PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Maison Henrique Pereira Fonseca Aawar

MODELO DE CLASSIFICAÇÃO PARA MANUTENÇÃO PREDITIVA EM EQUIPAMENTOS INDUSTRIAIS: PREVISÃO E ANÁLISE DE FALHAS

Belo Horizonte
Agosto de 2023

Maison Henrique Pereira Fonseca Aawar

MODELO DE CLASSIFICAÇÃO PARA MANUTENÇÃO PREDITIVA EM EQUIPAMENTOS INDUSTRIAIS: PREVISÃO E ANÁLISE DE FALHAS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Belo Horizonte

Agosto de 2023

SUMÁRIO

1. Introdução

A indústria atual é caracterizada por sua alta dinamicidade, onde a paralisação de um equipamento ou linha de produção pode acarretar prejuízos consideráveis às empresas. Por essa razão, é de extrema importância ter meios para identificar a possibilidade de falha em um equipamento, bem como compreender a natureza dessa falha, a fim de otimizar o processo produtivo.

2. Descrição do Problema e da Solução Proposta

A manutenção tem como objetivo preservar as condições de funcionamento e desempenho de uma máquina. Para garantir uma boa produtividade em processos de produção, como uma linha de montagem, é essencial contar com um planejamento adequado e uma execução eficiente da manutenção.

Existem quatro tipos principais de manutenção: corretiva, preventiva, programada e preditiva. A manutenção corretiva é acionada quando um equipamento apresenta um defeito inesperado, podendo exigir a substituição do mesmo em alguns casos. Esse processo geralmente resulta em custos superiores ao planejado, perda de eficiência e prejuízos decorrentes de possíveis paralisações.

A fim de evitar esses problemas, a manutenção preditiva tem ganhado destaque na indústria. Ela busca identificar antecipadamente se um equipamento está prestes a apresentar defeito. A instalação de sensores nos equipamentos permite obter informações que possam indicar o estado de funcionamento dos mesmos.

Dessa forma, é possível desenvolver um algoritmo de aprendizado de máquina capaz de identificar quando um equipamento está prestes a falhar e determinar qual o motivo da falha? Para responder a essa pergunta, será utilizado um modelo de aprendizado de máquina, especificamente o algoritmo de classificação multiclasse. Esse tipo de algoritmo é capaz de identificar diferentes tipos de falhas que podem ocorrer em um determinado equipamento.

3. Coleta de Dados

Os dados utilizados foram obtidos no Kaggle, disponíveis no link: https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-

classification, acessado em 26 de janeiro de 2023. Esses dados estão no formato CSV e compreendem um conjunto de 10.000 pontos de dados, organizados em linhas e com 10 características representadas em colunas, conforme descrito na tabela abaixo:

Nome do dataset: Predictive Maintenance Dataset Data Set
Descrição: Conjunto de dados sintético que reflete dados reais de manutenção preditiva encontrados na indústria.
Link: https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification

Nome do Atributo	Descrição	Tipo
UID	Identificador único varia de 1 a 10000	Integer
Product ID	ID do produto consistindo em uma letra L, M ou H para baixo (50% de todos os produtos), médio (30%) e alto (20%) como variantes de qualidade do produto e um número de série específico da variante	String
Type	Tipo de produto L, M ou H	String
Air temperature [K]	Temperatura do ar	Float
Process temperature [K]	Temperatura do processo	Float
Rotational speed [rpm]	Velocidade de rotação	Integer
Torque [Nm]	Torque	Float
Tool wear [min]	Desgaste da ferramenta	Integer
Target	Rótulo que indica se a máquina falhou ou não (0, 1)	Integer
Failure Type	Tipo de falha: Heat Dissipation Failure, Power Failure, Overstrain Failure, Tool Wear Failure, Random Failures e No Failure	String

4. Processamento/Tratamento de Dados

Para o tratamento dos dados, utilizou-se a linguagem de programação Python em conjunto com a biblioteca Pandas. Abaixo, apresenta-se uma análise geral da base de dados, onde é possível visualizar características como valores únicos, valores ausentes, tipos de dados e a contagem de cada variável.

```
check_df = pd.DataFrame({'type': df.dtypes, 'missing': df.isna().sum(),
                    'size': df.shape[0], 'unique': df.nunique()})
check_df['percentual_missing']=round(check_df['missing']/check_df['size'], 2)
```

	type	missing	size	unique	percentual_missing
UDI	int64	0	10000	10000	0.00
Product ID	object	0	10000	10000	0.00
type	object	0	10000	3	0.00
air_temperature	float64	0	10000	93	0.00
process_temperature	float64	0	10000	82	0.00
rotacional_speed	int64	0	10000	941	0.00
torque	float64	0	10000	577	0.00
tool_wear	int64	0	10000	246	0.00
target	int64	0	10000	2	0.00
failure_type	object	0	10000	6	0.00

As variáveis Product ID e UID foram removidas para simplificar o estudo, uma vez que não têm impacto no modelo, sendo apenas códigos de identificação.

Foi foi feito uma verificação na variável alvo “Target” para identificar possíveis divergências. A primeira inconsistência encontrada diz respeito à classificação de “No Failure”. Foi encontrado 9 máquinas classificadas como “Failure (1)” que, na verdade, deveriam ser classificadas como “No Failure (0)”. Nesse caso, não é possível afirmar com certeza se ocorreu uma falha ou não, portanto, esses equipamentos foram removidos.

```
check_ambiguous = df.loc[(df['target']==1) & (df['failure_type']=='No
Failure')].index
df.loc[check_ambiguous, check_target]
```

	target	failure_type
1437	1	No Failure
2749	1	No Failure
4044	1	No Failure
4684	1	No Failure
5536	1	No Failure
5941	1	No Failure
6478	1	No Failure
8506	1	No Failure
9015	1	No Failure

De acordo com a tabela abaixo, foi observado que algumas máquinas foram classificadas como “No Failure (0)”, quando deveriam ser classificadas como “Failure (1)”. Na base de dados, apenas 18 máquinas foram identificadas como Random Failures, caracterizadas por serem de natureza aleatória e imprevisível. E essas instâncias também foram removidas.

```
check_target = ['target','failure_type']
check_rnf = df.loc[df['failure_type'] == 'Random Failures'].index
df.loc[check_rnf,check_target]
```

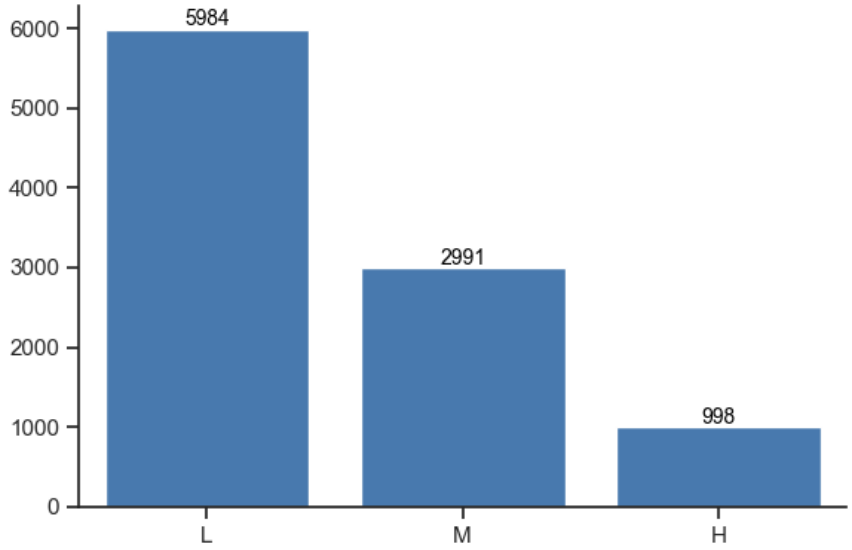
	target	failure_type
1221	0	Random Failures
1302	0	Random Failures
1748	0	Random Failures
2072	0	Random Failures
2559	0	Random Failures
3065	0	Random Failures
3452	0	Random Failures
5471	0	Random Failures
5489	0	Random Failures
5495	0	Random Failures
5509	0	Random Failures
5553	0	Random Failures
5639	0	Random Failures
6091	0	Random Failures
6913	0	Random Failures
6960	0	Random Failures
7488	0	Random Failures
7868	0	Random Failures

Após o tratamento realizado na base de dados, torna-se possível conduzir as análises e exploração dos dados com maior coerência. Vale ressaltar que todas as modificações feitas não tiveram um impacto significativo na estrutura geral da base de dados.

5. Análise e Exploração dos Dados

A análise exploratória de dados desempenha um papel essencial no processo de análise de dados, visando descobrir padrões e insights ocultos nos conjuntos de dados. Durante a análise exploratória, foi possível examinar a distribuição dos dados, identificar a presença de outliers e estabelecer possíveis relações entre as variáveis. Cada máquina no conjunto de dados possui um código ID que consiste em uma letra inicial representando o tipo de máquina, seguida por uma sequência numérica. Ao analisar o conjunto de dados, foi constatado que a maioria das máquinas pertence ao tipo L conforme demonstrado no Gráfico 1.

Gráfico 1 - Tipo de Produto



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

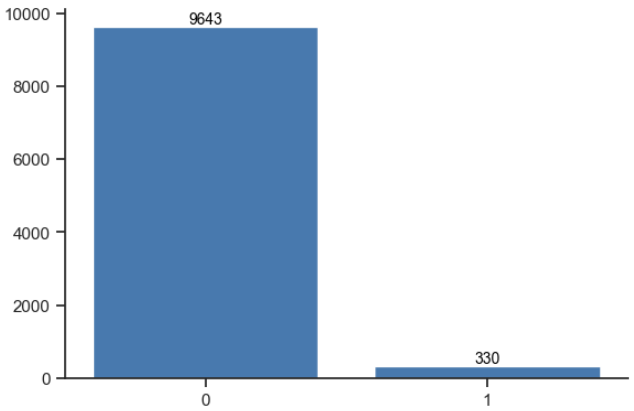
Outra observação relevante é a baixa ocorrência de falhas de máquinas em todo o conjunto de dados, em termos percentuais, a categoria "No Failure" representa 96,52% dos dados, enquanto a categoria "Failure" representa 3,48%.

Além disso, ao analisar o gráfico que apresenta as causas associadas a cada falha, identifica-se um desequilíbrio adicional. Isso significa que algumas causas têm uma presença muito maior do que outras no conjunto de dados, o que pode enviesar as análises e influenciar os resultados.

Portanto, é essencial realizar um balanceamento dos dados para lidar com o desequilíbrio observado, como evidenciado no Gráfico 2. A abordagem para resolver

essa questão será explorada nas próximas etapas do projeto, visando obter uma representação mais equilibrada das diferentes classes e evitar qualquer viés na análise e nos resultados.

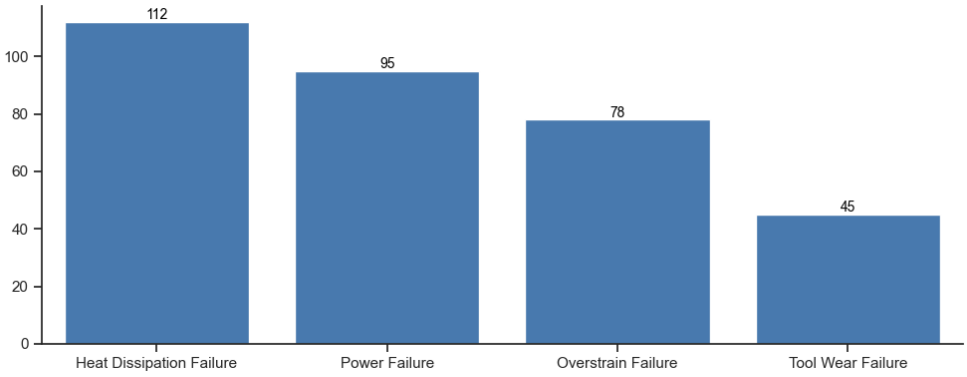
Gráfico 2 - Quantidade de falhas



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

O Gráfico 3 ilustra os tipos de falhas presentes no conjunto de dados são descritos da seguinte maneira: Heat Dissipation Failure (falha na dissipação de calor), Power Failure (falha de energia), Overstrain Failure (falha de sobrecarga) e Tool Wear Failure (falha de desgaste da ferramenta). Essas categorias representam os diferentes tipos de problemas ou eventos que podem ocorrer no contexto analisado.

Gráfico 3 - Quantidade do tipo de falhas



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

Ao examinar a Tabela 1 é possível detectar a presença de valores discrepantes em Rotational Speed e Torque, pois seus valores máximos estão consideravelmente

distantes do terceiro quartil. No entanto, para embasar de forma mais sólida essa observação, é necessário conduzir análises adicionais como o objetivo de compreender melhor essas informações.

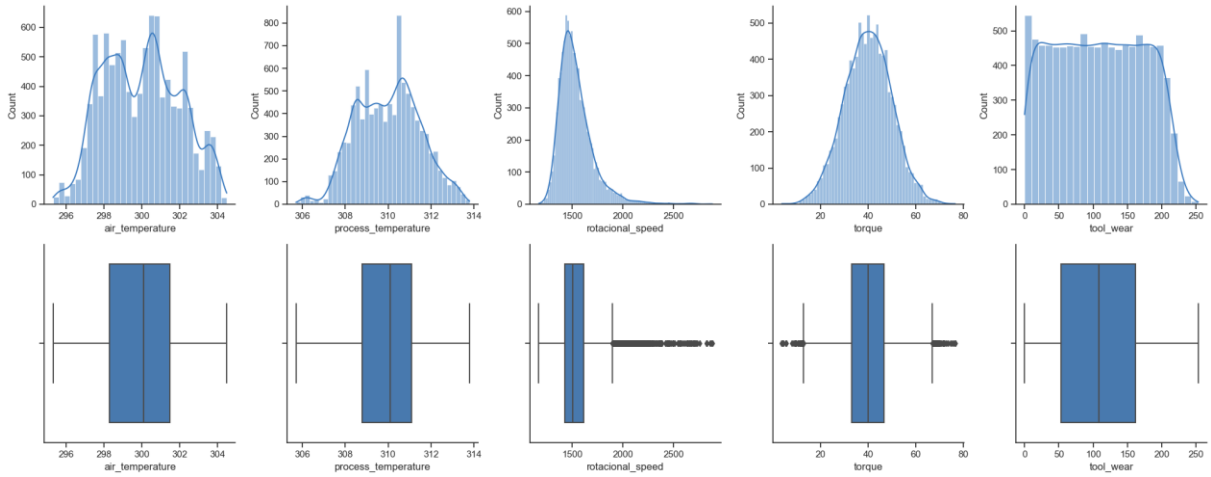
Tabela 1 – Estatística descritiva

	air_temperature	process_temperature	rotacional_speed	torque	tool_wear
count	9973.00	9973.00	9973.00	9973.00	9973.00
mean	300.00	310.00	1538.89	39.98	107.92
std	2.00	1.48	179.41	9.97	63.65
min	295.30	305.70	1168.00	3.80	0.00
25%	298.30	308.80	1423.00	33.20	53.00
50%	300.10	310.10	1503.00	40.10	108.00
75%	301.50	311.10	1612.00	46.70	162.00
max	304.50	313.80	2886.00	76.60	253.00

Fonte: Elaborado pelo autor com dados extraídos da base de dados.

O Gráfico 4 de boxplot destacam a presença de possíveis valores discrepantes no Torque e Velocidade de Rotação, conforme mencionado anteriormente. Embora esses valores discrepantes sejam observados, após considerar outros aspectos relevantes, decidi manter esses valores no conjunto de dados.

Gráfico 4 – Distribuição das variáveis

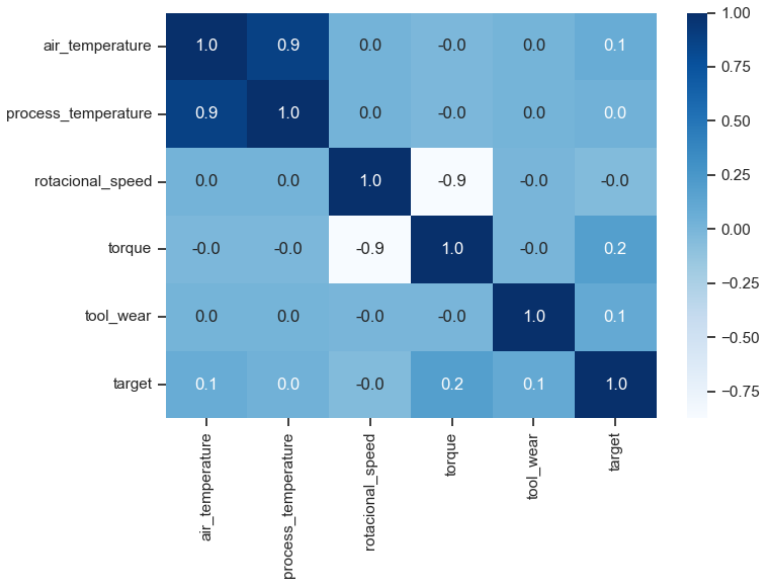


Fonte: Elaborado pelo autor com dados extraídos da base de dados.

Além disso, ao analisar o Gráfico 4 de histograma, é possível observar que a distribuição da Velocidade de Rotação é assimétrica positivamente. Isso significa que há uma tendência de concentração dos valores em direção aos valores mais altos, enquanto a cauda direita é mais longa. Essa assimetria pode indicar uma tendência ou padrão presente na variável de interesse.

No Gráfico 5 de correlação, é possível observar uma forte relação entre o torque e a velocidade de rotação. Além disso, constatamos uma alta correlação entre a temperatura do processo e a temperatura do ar. Essa ampla correlação sugere uma conexão significativa entre essas variáveis, indicando que mudanças na temperatura podem ter um impacto direto nas medições de potência, e vice-versa. A correlação entre o torque e a velocidade de rotação pode indicar uma relação direta de dependência entre essas grandezas.

Gráfico 5 – Correlação



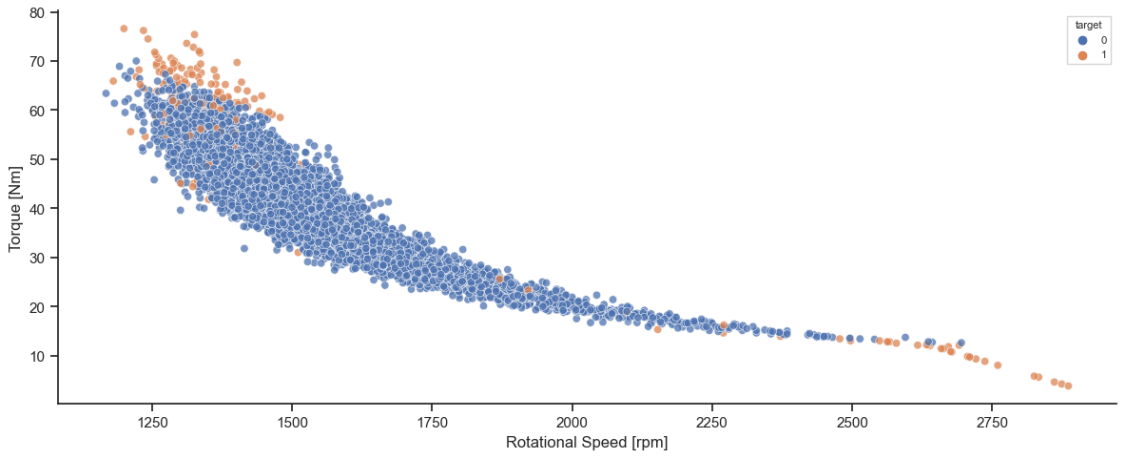
Fonte: Elaborado pelo autor com dados extraídos da base de dados.

Podemos observar prontamente que as falhas ocorrem para valores extremos de algumas características, ou seja, o maquinário falha tanto para os menores quanto para os maiores valores de torque e velocidade de rotação. Essa tendência é facilmente identificada no Gráfico 6, onde os pontos laranjas estão distantes nessas

regiões. Portanto, existe uma faixa de condições normais em que as máquinas operam, e acima ou abaixo dessa faixa, elas tendem a apresentar falhas.

Essa observação é relevante, pois indica que as máquinas são sensíveis a condições extremas de torque e velocidade de rotação. Esses pontos críticos representam uma região em que é necessário monitorar de perto o desempenho das máquinas e implementar medidas preventivas para evitar falhas. Essa compreensão dos limites de operação normal das máquinas é fundamental para estabelecer critérios de segurança e tomar decisões embasadas na prevenção de falhas e no aumento da confiabilidade do maquinário.

Gráfico 6 – Relação Torque x Rotacional Speed



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

Em resumo, a análise exploratória de dados permitiu identificar padrões, compreender a estrutura dos dados, fornecendo insights valiosos que sustentam as próximas etapas do projeto.

6. Preparação dos Dados para os Modelos de Aprendizado de Máquina

Após a análise exploratória, foi feito a preparção dos dados e, posteriormente, o treinamento dos modelos de aprendizado de máquina. Para tornar os dados adequados para os algoritmos que serão utilizados, foi realizado transformações, na coluna “Type” e “Failure Type”. A transformação consistiu na aplicação de uma codificação de rótulo nas colunas categóricas. A codificação seguiu conforme código a seguir:

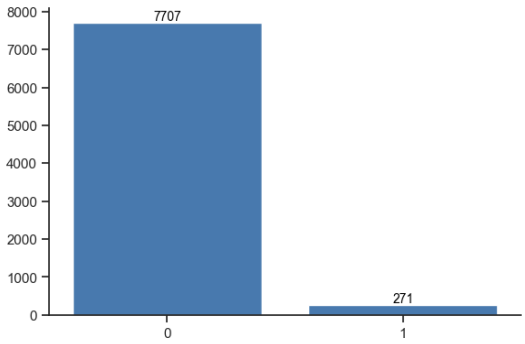
```
type_dict = {'L': 0, 'M': 1, 'H': 2}
cause_dict = {'No Failure': 0, 'Power Failure': 1, 'Overstrain Failure': 2,
              'Heat Dissipation Failure': 3, 'Tool Wear Failure': 4}
df1['type'].replace(to_replace=type_dict, inplace=True)
df1['failure_type'].replace(to_replace=cause_dict, inplace=True)
```

Essa codificação de rótulo permite que as informações categóricas sejam representadas por valores numéricos, tornando-as compatíveis com os algoritmos de análise e modelagem que serão aplicados aos dados. Após realizar a codificação como descrito anteriormente, é possível agora separar os dados em conjuntos de treinamento e teste, conforme demonstrado no código a seguir:

```
X = df1.drop(['target', 'failure_type'], axis=1)
y = df1['failure_type']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42)
```

Ao realizar a separação dos dados em conjuntos de treinamento e teste, é fundamental considerar o desbalanceamento identificado no Gráfico 7. Esse desequilíbrio é evidenciado pela proporção maior de máquinas que não falharam, representadas pela classe "0" na variável "Target".

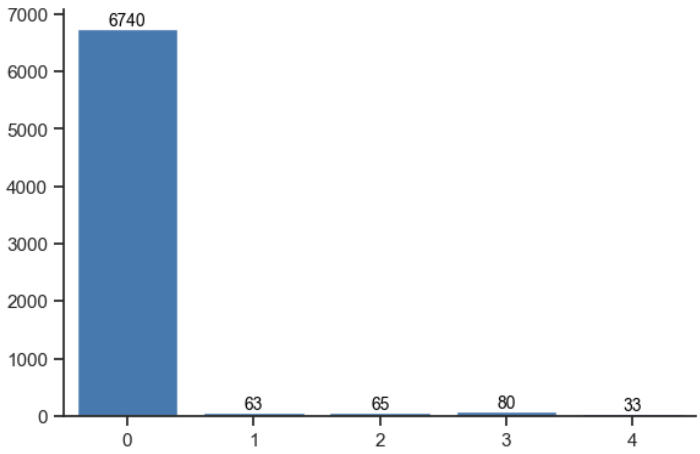
Gráfico 7 - Distribuição das classes – Variável Target



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

No Gráfico 8, é possível observar uma maior quantidade de máquinas sem falhas na variável "Failure Type", que identifica os tipos de falhas registrados.

Gráfico 8 - Distribuição das classes – Variável Failure Type



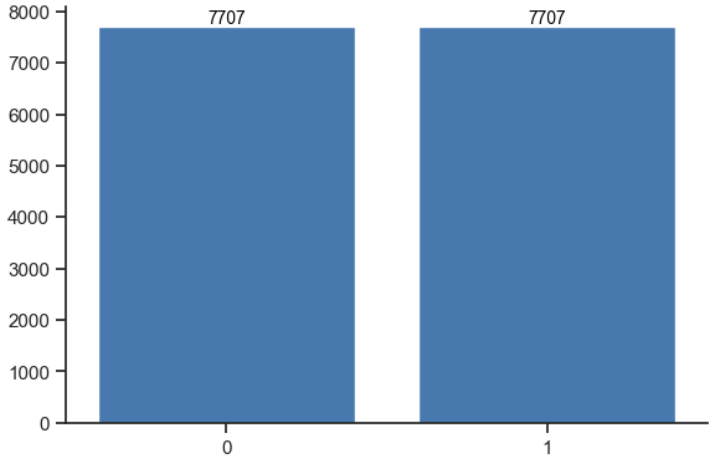
Fonte: Elaborado pelo autor com dados extraídos da base de dados.

Para lidar com o desbalanceamento, foi adotada a técnica de oversampling, que consiste em aumentar as amostras da classe minoritária. Nesse caso, foram testados os métodos SMOTE, ADASYN, BorderlineSMOTE, SMOTEN e RandomOverSampler. Abaixo o código correspondente:

```
def balance_data_binary(X_train, y_train, method='SMOTE'):
    if method == 'SMOTE':
        smote = SMOTE(random_state=42)
        X_train_balanced, y_train_balanced = smote.fit_resample(X_train,
y_train)
    elif method == 'ADASYN':
        adasyn = ADASYN(random_state=42)
        X_train_balanced, y_train_balanced = adasyn.fit_resample(X_train,
y_train)
    elif method == 'BorderlineSMOTE':
        blsmote = BorderlineSMOTE(random_state=42)
        X_train_balanced, y_train_balanced = blsmote.fit_resample(X_train,
y_train)
    elif method == 'SMOTEN':
        smoten = SMOTEN(random_state=42)
        X_train_balanced, y_train_balanced = smoten.fit_resample(X_train,
y_train)
    elif method == 'RandomOverSampler':
        rdsampler = RandomOverSampler(random_state=42)
        X_train_balanced, y_train_balanced = rdsampler.fit_resample(X_train,
y_train)
    else:
        X_train_balanced, y_train_balanced = X_train, y_train
    return X_train_balanced, y_train_balanced
```

Dentre os métodos testados, o SMOTEN obteve melhor desempenho, conforme observado no Gráfico 9 é possível perceber o balanceamento dos dados da variável "Target" após a aplicação desse método.

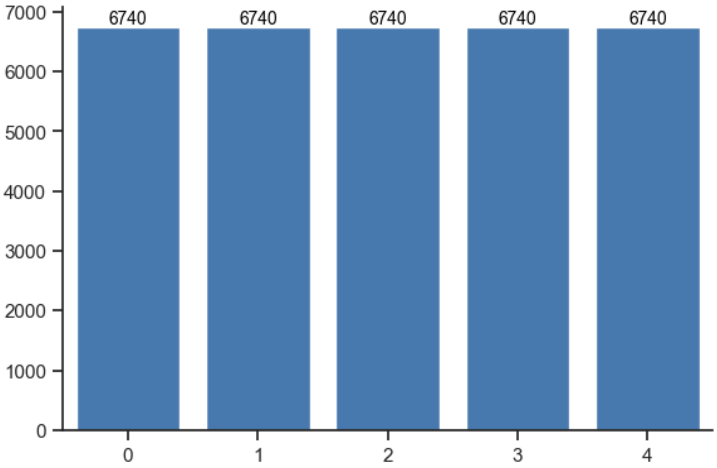
Gráfico 9 - Distribuição das classes - Variável Target



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

No Gráfico 10, também é evidente o balanceamento dos dados da variável “Feilure Type”, foi utilizado o método “SMOTE”. Esse balanceamento é fundamental para evitar o viés introduzido pelo desequilíbrio das classes presentes nos dados.

Gráfico 10 - Distribuição das classes - Variável Failure Type



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

Em resumo, o processo de pré-processamento desempenha um papel crucial antes da aplicação dos modelos de aprendizado de máquina. O código completo encontra-se no final do projeto, com o respectivo link disponibilizado no repositório do Github.

7. Aplicação de Modelos de Aprendizado de Máquina

Para o problema de classificação binária, cujo objetivo é identificar se uma máquina irá falhar ou não, foi feito teste com os seguintes algoritmos: SVC, Regressão Logística, Random Forest, XGBoost, LightGBM e Gradient Boosting.

Para classificar o tipo de falha de uma máquina, foram utilizados modelos de classificação multiclasse. Os mesmos algoritmos foram empregados, porém, o que difere do modelo de classificação binária são os parâmetros específicos que foram utilizados. Abaixo o código utilizado para treinar os modelos de classificação multiclasse:

```
def train_models(classifiers, X_train, y_train):
    trained_models = {}
    for clf_name, clf in classifiers.items():
        pipeline = make_pipeline(RobustScaler(), clf)
        pipeline.fit(X_train, y_train)
        trained_models[clf_name] = pipeline
    return trained_models
trained_models = train_models(classifiers, X_train_m_balanced,
y_train_m_balanced)
```

Para avaliar os modelos, foram utilizadas as seguintes métricas de avaliação: Acurácia (Accuracy), Precisão (Precision), Recall (Revocação ou Sensibilidade), F1-score e Área sob a curva ROC (ROC-AUC). Cada uma dessas métricas fornece uma perspectiva diferente sobre o desempenho do seu modelo de classificação. Enquanto a acurácia é uma métrica geral, as métricas de precisão, recall e F1-score são especialmente úteis quando há desequilíbrio entre as classes. O ROC-AUC avalia a habilidade do modelo em fazer distinções corretas entre as classes.

8. Avaliação dos Modelos de Aprendizado de Máquina e Discussão dos Resultados

O resultado do treinamento dos modelos para classificação binária é apresentado abaixo, considerando todas as métricas de avaliação:

	Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
0	SVC	0.95	0.67	0.80	0.71	0.80
1	Logistic Regression	0.88	0.55	0.68	0.56	0.68
2	Random Forest	0.98	0.92	0.75	0.81	0.75
3	XGBoost	0.98	0.86	0.85	0.86	0.85
4	LightGBM	0.99	0.93	0.89	0.91	0.89
5	Gradient Boosting	0.98	0.85	0.88	0.86	0.88

Pode-se observar que todos os modelos apresentam resultados razoáveis, mas alguns modelos têm desempenho superior em relação a outros. O modelo XGBoost, LightGBM e Gradient Boosting apresentaram um desempenho geral mais elevado em relação às outras abordagens, com F1-Score acima de 0.81, o que indica uma boa capacidade de classificação das instâncias do conjunto de dados. Com base nos resultados anteriores, o modelo LightGBM apresentou o melhor desempenho, conforme o relatório de classificação abaixo:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1936
1	0.87	0.78	0.82	59
accuracy			0.99	1995
macro avg	0.93	0.89	0.91	1995
weighted avg	0.99	0.99	0.99	1995

A análise do relatório de classificação apresenta uma avaliação abrangente do desempenho do modelo LightGBM. A acurácia global do modelo atingiu 99%, isso significa que a maioria das previsões está correta, tanto para a classe "No Failure" quanto para a classe "Failure". Os resultados ressaltam o sólido desempenho geral do modelo LightGBM na classificação das instâncias do conjunto de dados. O modelo demonstrou resultados promissores tanto para as máquinas com falhas quanto para as sem falhas, validando sua eficácia na tarefa de classificação.

A matriz de confusão indica que o modelo apresenta um excelente desempenho geral, com alta acurácia de aproximadamente 99%.

A precisão para a classe "Failure" é de 87%, o que significa que cerca de 87% das amostras classificadas como "Failure" pelo modelo realmente pertencem a essa classe. O recall, para a classe "Failure" é de 78%, indicando que o modelo é capaz de identificar corretamente cerca de 78% das amostras reais da classe "Failure".

A especificidade é extremamente alta para a classe "No Failure" (99%), o que significa que o modelo é excepcionalmente bom em identificar corretamente amostras da classe "No Failure". O F1-Score de aproximadamente 82% indica um equilíbrio sólido entre precisão e recall, sugerindo que o modelo está fazendo um bom trabalho em ambas as classes.

Gráfico 11 – Matriz de Confusão – Variável Target

Fonte: Elaborado pelo autor com dados extraídos da base de dados.

Em resumo, o modelo LightGBM demonstra um desempenho excepcional, com alta precisão, recall e acurácia. Isso pode ser indicativo de um modelo bem treinado e apropriado para a tarefa em questão.

Para os modelos de classificação multiclasse abaixo é apresentado os resultados de treinamento:

	Model	Accuracy	Precision	Recall	F1-score
0	SVC	0.91	0.45	0.83	0.54
1	Logistic Regression	0.88	0.39	0.85	0.48
2	Random Forest	0.97	0.60	0.77	0.65
3	XGBoost	0.98	0.66	0.79	0.71
4	LightGBM	0.98	0.66	0.80	0.71
5	Gradient Boosting	0.95	0.58	0.87	0.66

Ao analisar esses resultados, podemos observar o desempenho de cada modelo multiclasse com base em diferentes métricas de avaliação. Os modelos XGBoost e LightGBM apresentaram acurácias de 98%, destacando-se na classificação, mas com diferenças nas métricas de precisão, recall e F1-Score.

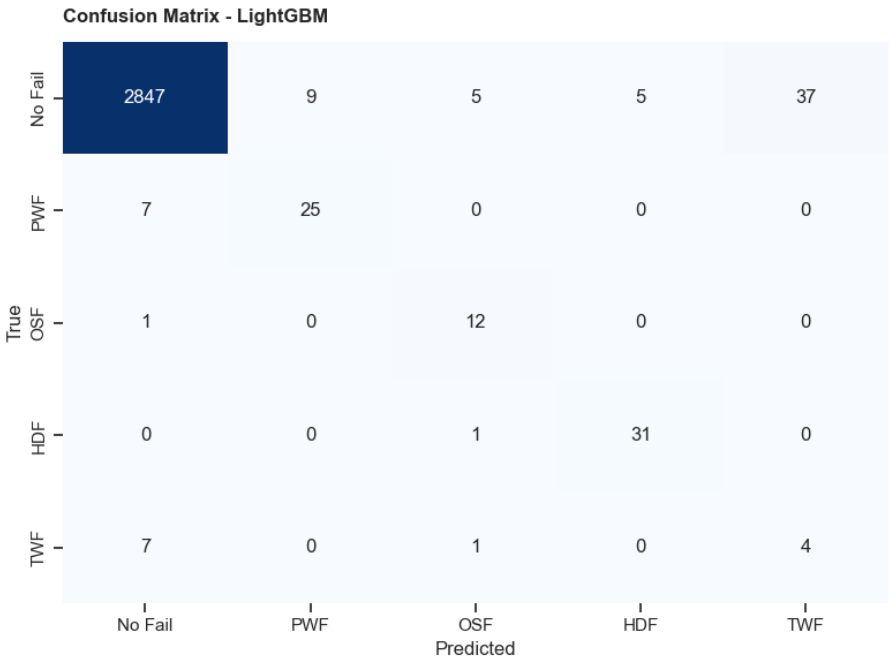
O modelo LightGBM apresentou o melhor desempenho após o treinamento com hiperparâmetros. A avaliação do relatório de classificação demonstra que para a classe "0", que representa uma categoria, o modelo atingiu uma precisão de 99%,

recall de 98% e F1-Score de 99%. Isso indica que o modelo possui alta precisão em identificar corretamente essa categoria e é capaz de recuperar eficazmente as instâncias verdadeiramente pertencentes a essa classe.

	precision	recall	f1-score	support
0	0.99	0.98	0.99	2903
1	0.74	0.78	0.76	32
2	0.63	0.92	0.75	13
3	0.86	0.97	0.91	32
4	0.10	0.33	0.15	12
accuracy			0.98	2992
macro avg	0.66	0.80	0.71	2992
weighted avg	0.99	0.98	0.98	2992

Com base na matriz de confusão Gráfico 12, o modelo LightGBM apresenta um desempenho geral sólido, com uma acurácia de aproximadamente 97.00%. As métricas de precisão macro, recall macro e F1-Score macro indicam que o modelo está lidando razoavelmente bem com as diferentes classes, mas pode haver espaço para melhorias, especialmente nas classes PWF, OSF e TWF, onde as métricas são mais baixas. Por outro lado, o modelo apresenta um desempenho sólido na classe majoritária "No Failure".

Gráfico 12 – Matriz de Confusão – Variável Failure Type



Fonte: Elaborado pelo autor com dados extraídos da base de dados.

9. Conclusão

Em conclusão, o projeto abordou a importante questão da manutenção industrial, visando melhorar a eficiência e a produtividade por meio da implementação de manutenção preditiva. Através da análise detalhada dos diferentes tipos de manutenção e das vantagens da abordagem preditiva, o projeto buscou desenvolver um algoritmo de aprendizado de máquina capaz de identificar potenciais falhas em equipamentos industriais, permitindo a intervenção antes que ocorram paralisações não planejadas.

A avaliação dos modelos de aprendizado de máquina, com destaque para o algoritmo LightGBM, demonstrou resultados promissores. O relatório de classificação exibiu um desempenho sólido, com alta precisão, recall e acurácia. Isso indica que o modelo é capaz de identificar corretamente as falhas e as não falhas, demonstrando sua eficácia na tarefa de classificação.

Já os resultados obtidos do modelo de classificação multiclasse, mostraram que o modelo LightGBM é especialmente adequado para a tarefa em questão. Sua capacidade de distinguir entre diferentes classes de falha e de não falha. A análise da matriz de confusão destacou essas áreas, sugerindo possíveis ajustes para otimizar ainda mais o modelo, aumentando sua precisão em todas as categorias.

Em resumo, o projeto demonstrou a viabilidade e a eficácia da abordagem de manutenção preditiva por meio da implementação de algoritmos de aprendizado de máquina.

Foram identificadas algumas limitações no estudo. A disponibilidade limitada de dados, tanto em termos de qualidade quanto de quantidade, pode comprometer o desempenho do modelo. Além disso, a existência de viés nos dados pode resultar em previsões imprecisas ou enviesadas.

Trabalhos futuros incluem aprimorar os dados através da coleta diversificada, explorar a engenharia de recursos para padrões mais complexos e otimizar o modelo com regularização e ajuste de hiperparâmetros.

10. Links

Repositório Github: https://github.com/maisonhenrique/ProjetoIntegrado