Aprendizagem Computacional
Machine Predictive Maintenance Classification
Relatório do projeto

Gonçalo Tavares de Bastos nº2020238997
Leonardo Cordeiro Gonçalves nº2020228071

May 2023

1 Introdução ao problema

Neste projeto, iremos trabalhar com um conjunto de dados de manutenção preditiva obtido do Kaggle, que reflete a manutenção preditiva do mundo real encontrada na indústria. O conjunto de dados contém 10000 pontos de dados, cada um com 14 características, e o objetivo do projeto é prever se houve ou não uma falha e prever o tipo de falha.

Para alcançar este objetivo, iremos aplicar técnicas de aprendizagem automática que aprendemos ao longo do semestre, como Máquinas de Vetor de Suporte (SVM) e Redes Neurais (NN). Iremos também escolher um terceiro método de aprendizagem automática para usar e justificar a nossa escolha.

Nas secções seguintes, iremos começar por importar e analisar os dados, seguido pela concepção e execução de experiências para avaliar o desempenho dos nossos métodos escolhidos. Finalmente, apresentaremos e discutiremos os nossos resultados.

2 Analise dos Dados

2.1 Descrição do Dataset

O conjunto de dados consiste em 10000 pontos de dados armazenados como linhas com 14 recursos em colunas:

- **UID:** identificador unico que varia de 1 a 10000
- **productID:** L, M ou H para variantes de qualidade de produto baixa (50%) de todos os produtos), média (30%) e alta (20%) e um número de série específico da variante
- **Type:** Obtido do Product ID, removendo o numero de série (ou seja, L, M ou H), pois este não é relevante para a maquina ter ou não falhas.
- **temperatura do ar [K]:** gerada usando um processo de random walk posteriormente normalizado para um desvio padrão de 2 K em torno de 300 K
- **temperatura do processo [K]:** gerada usando um processo de random walk normalizado para um desvio padrão de 1 K, adicionado à temperatura do ar mais 10 K.
- **velocidade de rotação [rpm]:** calculada a partir da potência de 2860 W, sobreposta com um ruído normalmente distribuído.
- **torque [Nm]:** os valores de torque são normalmente distribuídos em torno de 40 Nm com $\sigma = 10$ Nm e sem valores negativos.
- **desgaste da ferramenta [min]:** As variantes de qualidade H / M / L adicionam 5/3/2 minutos de desgaste da ferramenta à ferramenta usada no processo.
- **Target:** 0 para No Failure, 1 para Failure, label que vamos usar para a classificação binária.
- **Failure Type:** No Failure, Heat Dissipation Failure, Power Failure, Overstrain Failure, Tool We, Failure, Random Failures, label que vamos usar na classificação multi-classe.

2.2 Compreensão e limpeza dos dados

Depois de importar o dataset, para entendermos melhor os dado com que estamos a trabalhar começamos por verificar se cada entrada era única e se não haviam valores duplicados (*df.nunique()*); fizemos isto de forma simples verificando se o numero de valores únicos para o Product ID correspondia ao numero de amostras. Verificamos também se haviam valores nulos (*df.isnull().sum()*) e ainda o tipo de dados em cada coluna (*df.info()*), e os valores unicos das mesmas. Então para uma primeira analise chegamos a conclusao de que:

- Existem 10000 dados.
- Não existem missing values.
- 3 Colunas tem dados categoricos, são elas, Type, Product ID and Failure Type

Mais concretamente quanto a coluna target 'Failure Type', que esta tem Valores (categoricos como ja referido) unicos:: 'No Failure', 'Power Failure', 'Tool Wear Failure','Overstrain Failure', 'Random Failures' e 'Heat Dissipation Failure'. Fizemos ainda a contagem das ocorrências destes obtendo:

- No Failure: 9652
- Heat Dissipation Failure: 112
- Power Failure: 95
- Overstrain Failure: 78
- Tool Wear Failure: 45

- Random Failures: 18

Verificamos logo aqui que haveria um grande desiquilibrio entre a classe 'No Failure' e as restantes, este desiquilibrio no dataset também se pode comprovar na percentagem mínima de ocorrência de qualquer tipo de falha, mais concretamente 0.0339 para 0.9661 para 'No Failure', como podemos observar na figura 1. Lidaremnos com isto mais a frente.

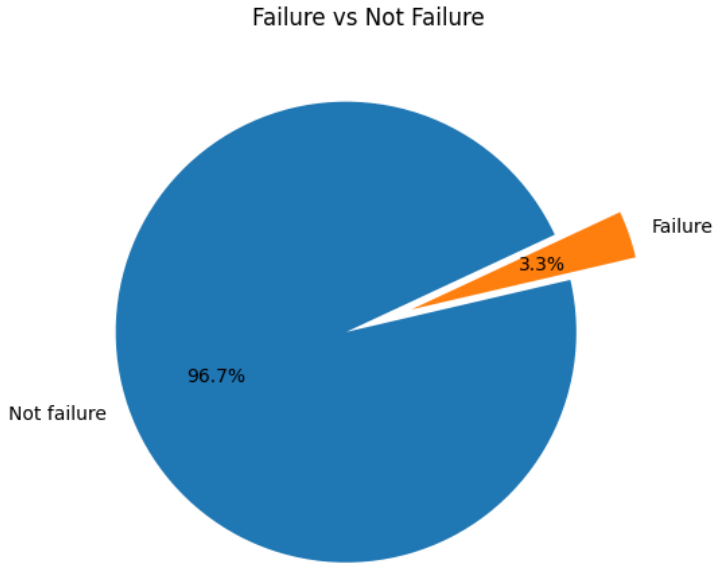


Figura 1: Failure vs Not Failure

2.2.1 Colunas UID,Product ID and Type

Antes de avançarmos também tivemos de tratar da coluna UID, pois esta não tem qualquer importância no estudo que estamos a fazer, no que toca a coluna Product ID esta não carrega mais informação para alem do Type, então também podemos descarta-la (usando *df.drop()*).

2.2.2 Anomalias nas colunas Target

Nesta secção, observamos distribuição do target para identificar eventuais desequilíbrios com o Failure Type e corrigi-los antes de fazermos o train test split. Verifica-se uma inconsistência entre o Target e o Failure Type, mais concretamente na classe 'No Failure', onde temos 9 valores 'Failure' no target e não no 'Failure Type' como não nos é possivel saber se são ou não 'Failure' removemos essas 9 linhas. Outra anomalia notada foi de que quando a falha é 'Random Failure' a coluna Target esta a 0 quando deveria estar a 1, isto ocorre em 18 observações e decidimos remove-las. No total removemos 27(0.27%) amostras do DataSet: pela ambiguidade entre Failure e Not Failure nas tabelas 'Target' e 'Failure Type'.

3 Exploratory Data Analysis (EDA)

3.1 Correlation

Primeiro usamos pairplot() para criar uma grade de gráficos de dispersão e histogramas, onde cada linha e coluna representa uma variável diferente no DataFrame, e a diagonal mostra um histograma dessa variável. Os pontos nos gráficos de dispersão serão codificados por cores com base na coluna 'Target', permitindo a comparação visual das relações entre diferentes pares de variáveis para cada categoria de destino, conseguimos concluir:

- Torque está bastante correlacionado com O Rotational Speed
- Process Temperature e Air Temperature também estão bastante correlacionados
- Conseguimos ainda ver que os Failure ocorrem nos valores extremos das features.

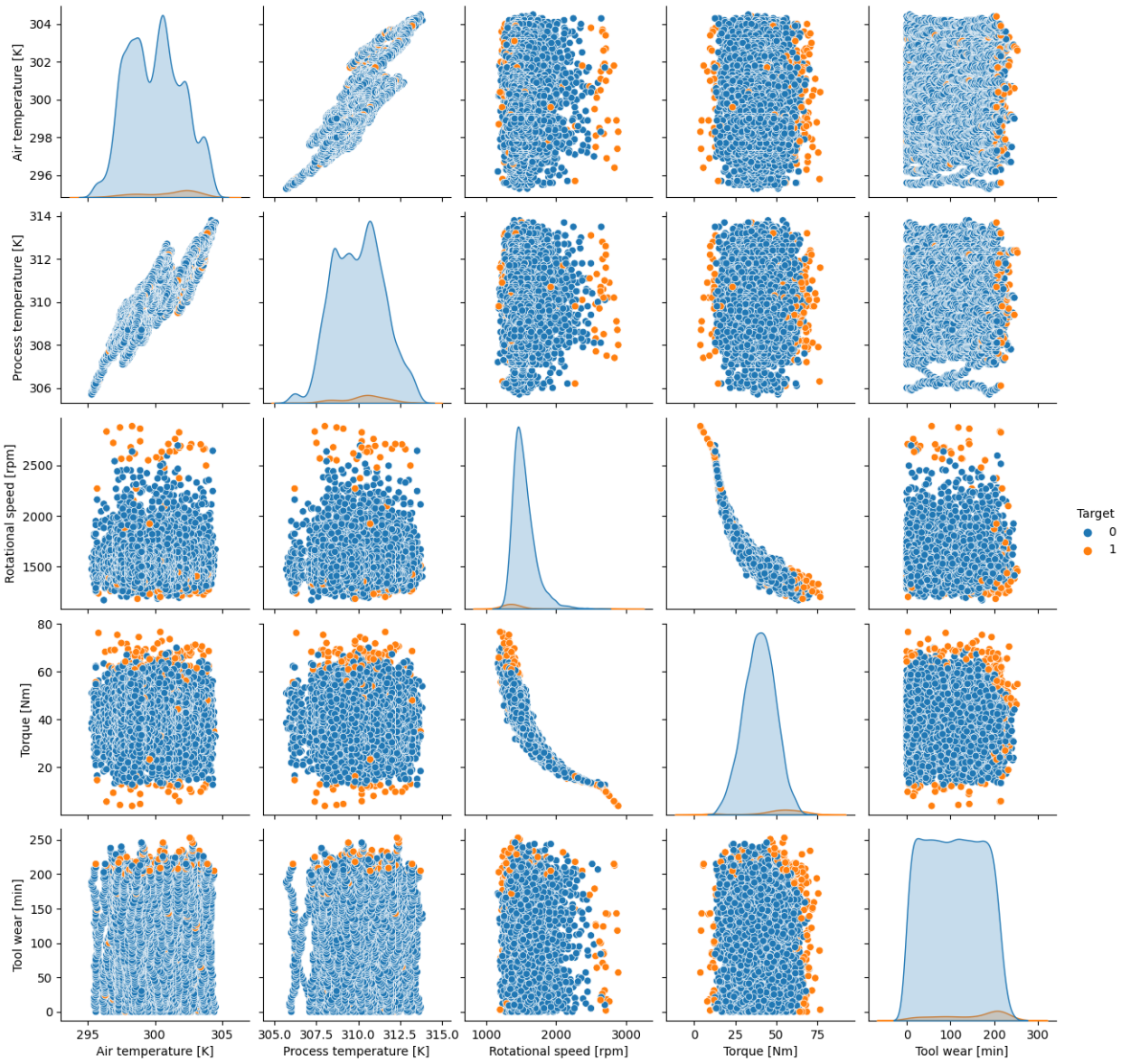


Figura 2: Correlação usando pairplot

Então de forma a analisar mais detalhadamente estes a correlação entre Torque e Rotational Speed, criamos uma figura com 2 subplots, o primeira subplot apresenta um scatter plot da relação entre a

velocidade de rotação e o torque, onde cada ponto é colorido de acordo com o tipo de falha correspondente, o segundo subplot apresenta o mesmo scatter plot, mas apenas para os casos em que houve falha (Target = 1), também coloridos pelo tipo de falha. O resultado esta na figura 3.

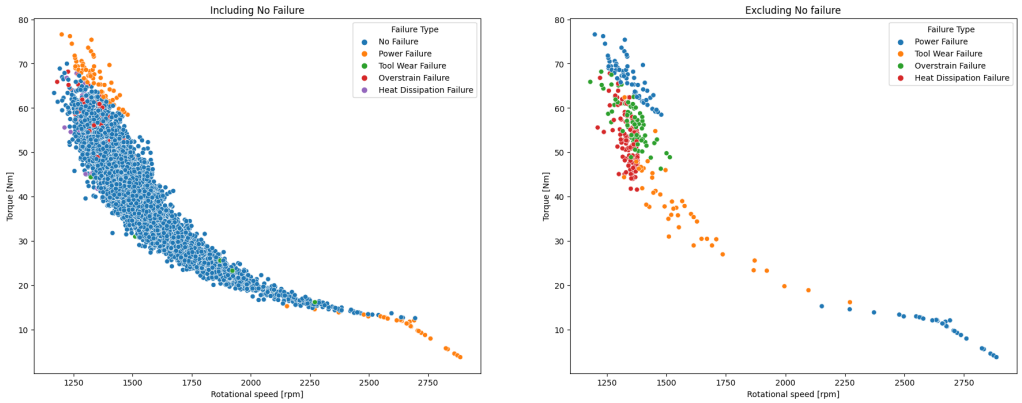


Figura 3: Torque e Rotational Speed

A figura 4 temos o mesmo processo para o Process Temperature e Air Temperature.

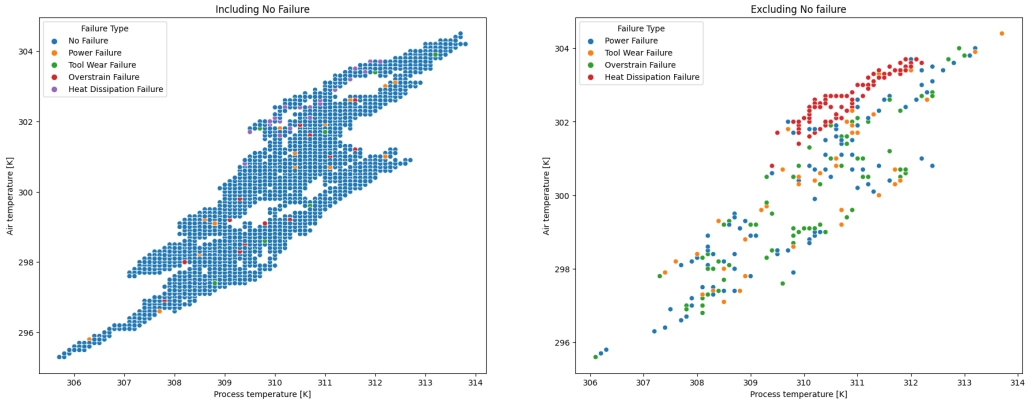


Figura 4: Process Temperature e Air Temperature

Usamos também um correlation heatmap (fig 5) para analisar as correlações entre as variáveis, e como mencionado anteriormente, há uma alta correlação entre a temperatura do processo e a temperatura do ar, e entre a velocidade de rotação e o torque.

3.2 Valores discrepantes

O objetivo desta secção é determinar se o conjunto de dados possui quaisquer valores discrepantes (outliers), pois estes podem influenciar negativamente os resultados dos modelos. Para isso primeiro analisamos usando a função describe() que no 'Rotational speed' e no 'torque' havia alguma discrepância entre a media e os valores máximos e com isto conseguimos prever a presença de valores discrepantes, para confirmarmos esta suposição utilizamos boxplots, histograms and Scatter plots (Figs 6-10) para analisar mais detalhadamente as distribuições, e verificamos a nossa suposição principalmente para o 'Rotational Speed' como se pode ver pela Fig6 é nitida a presença de valores atipicos. Lidaremos com estes valores discrepantes no pre-processamento.

Figura 5: Correlação usando heatmap

4 Pre-Processamento do dados

4.1 Codificação dos dados

Utilizamos o Ordinal Encoding para transformar variáveis categóricas Type e Failure Type em numéricas.

4.2 Escalonamento dos dados

Como verificado na secção de EDA existe uma presença notória de outliers tanto para o torque como para o Rotational speed, valores extremos que podem afetar negativamente a performance dos modelos, então nesta secção vamos fazer um scaling das features do Dataset. Os algoritmos de ML usam medidas para encontrar relações entre os dados, logo estes devem ser normalizados para que todos os pontos contribuam corretamente para análise, isto aumenta os desempenhos dos modelos Reduz a influência dos Outliers.

Utilizamos para fazer o escalonamento do 'Rotational speed' e do 'Torque' primeiramente utilizamos o MinMaxScaller pois este leva em conta o valor mínimo e máximo de cada feature para realizar o escalonamento. Isso significa que, mesmo que um outlier esteja presente no conjunto de dados, ele ainda será levado em conta no cálculo dos valores mínimo e máximo, o que ajuda a manter a

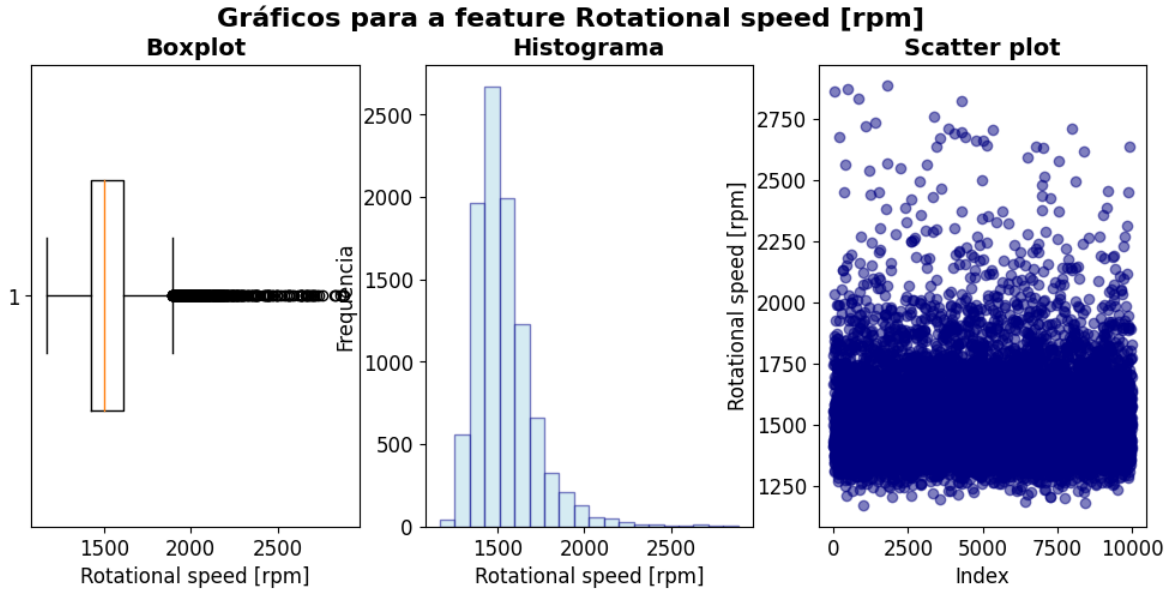


Figura 6: Distribuição para Rotational Speed

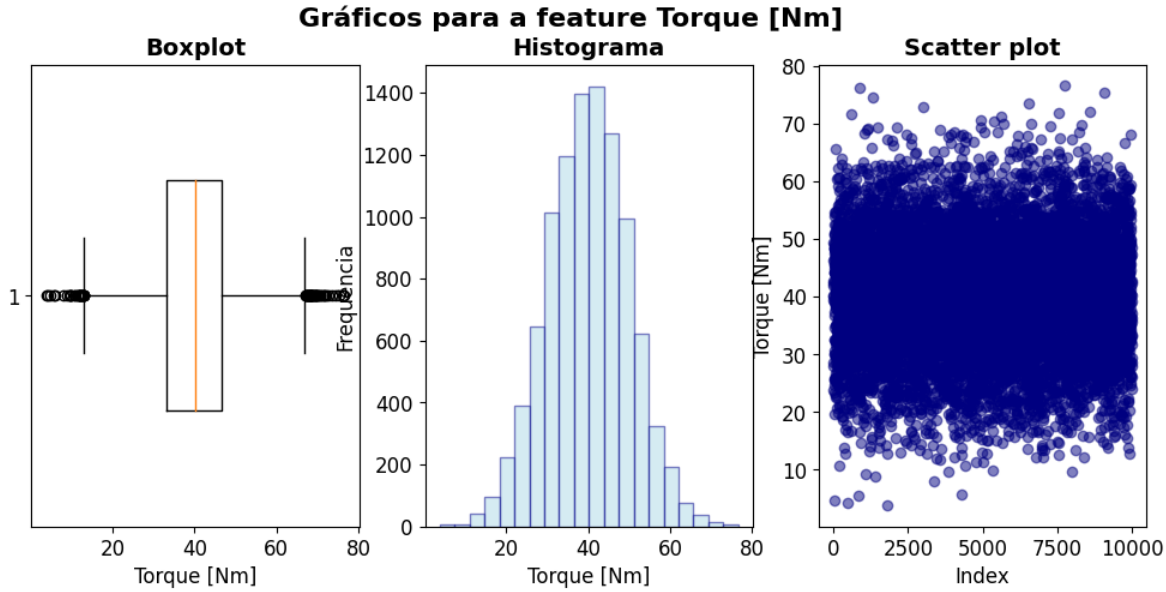


Figura 7: Distribuição para Torque

representatividade do conjunto de dados. Ainda assim verificamos que o RobustScaler se comportava ligeiramente melhor na presença dos outliers e assim acabamos por usa-lo.

Para a normalização dos restantes utilizamos o StandarScaler vimos que foi o que se comportou melhor para estes dados que teem uma distribuição normal.

4.3 Resampling

A necessidade de fazer resampling em bases de dados desbalanceadas surge quando a proporção de observações de uma determinada classe é significativamente menor do que a proporção de outra classe. Como mencionado na secção 2.2 estamos perante essa mesma situação, e pode afetar negativamente o desempenho dos modelos.

Com um dataset é desequilibrado, os modelos tendem a apresentar um desempenho inferior na

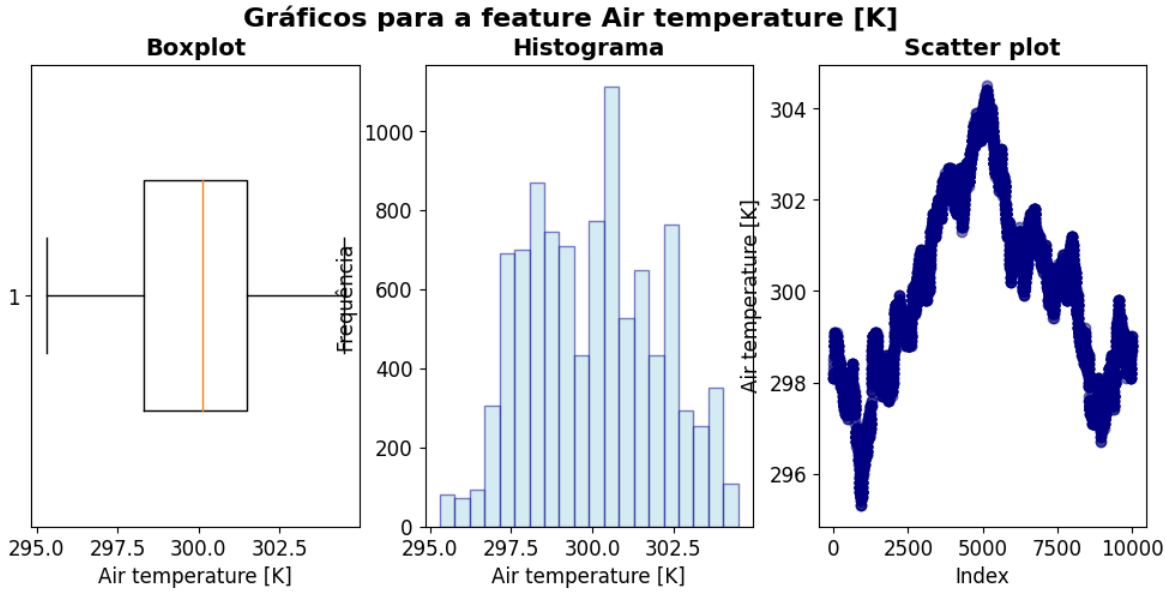


Figura 8: Distribuição para Air Temperature

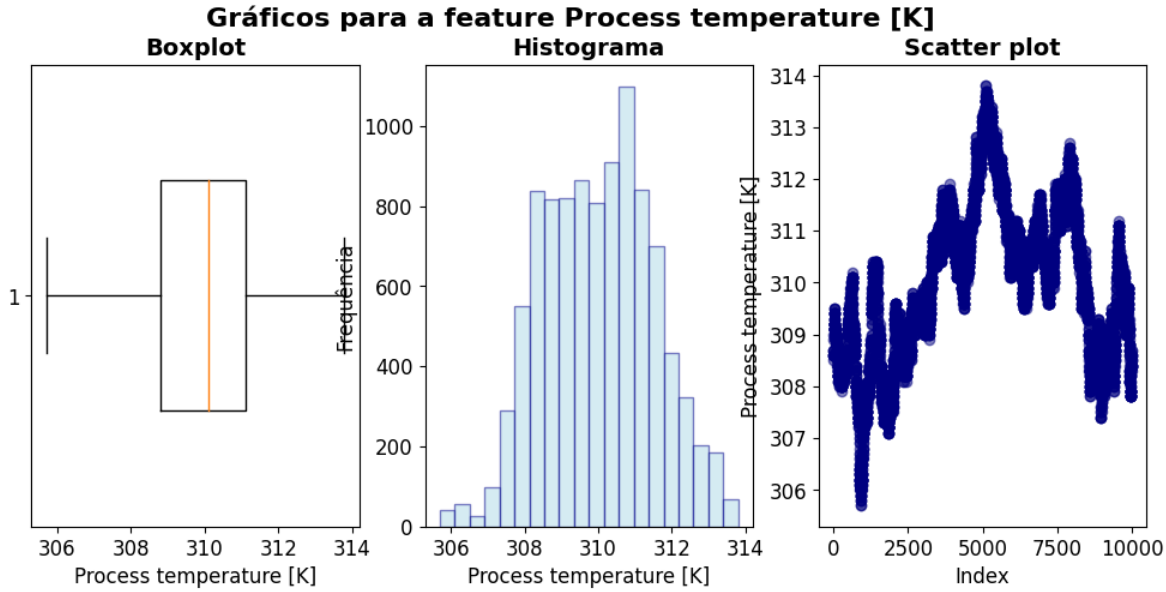


Figura 9: Distribuição para Process Temperature

classe minoritária,no nosso caso os tipos de falhas, porque o modelo pode ser tendencioso em direção à classe majoritária ('No Failure'), devido à alta frequência de observações dessa classe. Isso pode levar a um modelo que apresenta alta precisão na classe majoritária, mas baixa precisão na classe minoritária, que é a classe de interesse.

Para mitigar esse problema, iremos fazer resampling dos dados. O resampling é uma técnica que envolve a modificação da distribuição das classes na base de dados, para equilibrar a proporção de observações de cada classe. Existem duas formas principais de fazer resampling: oversampling e undersampling.

Criamos uma função que será usada na próxima secção em que iremos avaliar o desempenho de vários modelos, testando também vários tipos de oversamplers, usando a técnica de validação cruzada. E desta forma escolhemos e aplicamos a técnica de resampling que melhor resultados teve quando combinada com os modelos.

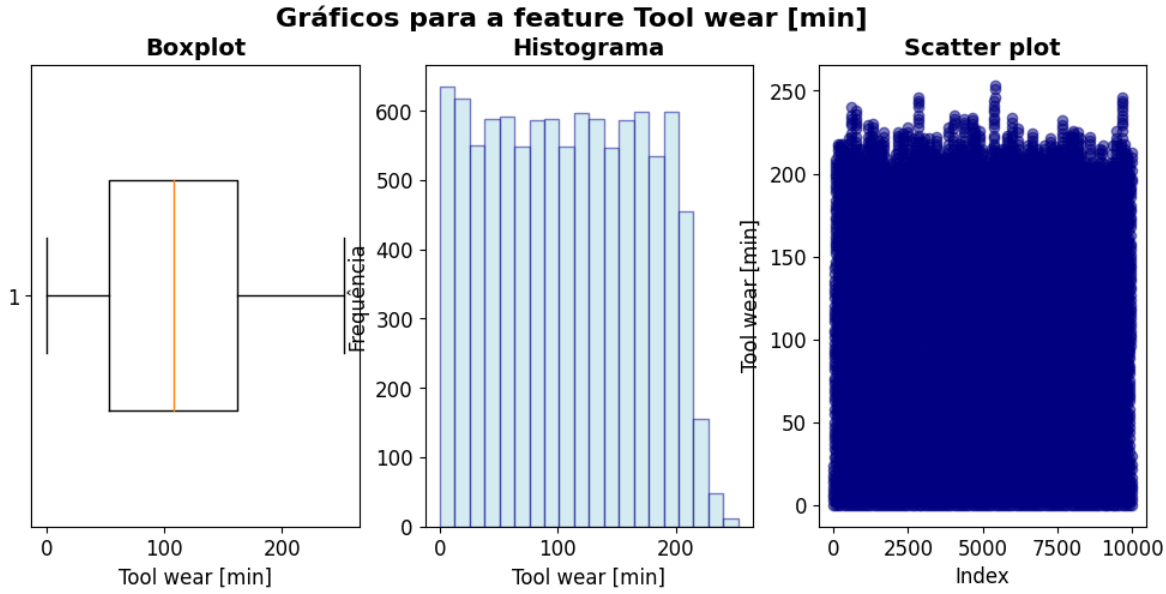


Figura 10: Distribuição para Tool Wear

5 Cenario A- Binary classification

Nesta seção o objetivo é classificação binária do conjunto de dados a fim de prever se haverá ou não falha da máquina.

Como foi dito na seccao 4.3, criamos uma função 'cross_validation' e usamo-la para avaliar o desempenho de vários modelos, testando também vários tipos de oversamplers, usando a técnica de validação cruzada. A função aceita uma lista de modelos, uma lista de oversamplers, o conjunto de dados completo (X e y), o número de dobras de validação cruzada (cv) e a métrica de avaliação (scoring) como entrada e retorna um DataFrame com os resultados da validação cruzada para todos os modelos e oversamplers testados. O dataframe esta na figura 11.

	accuracy_mean	accuracy_std	precision_mean	precision_std	recall_mean	recall_std	f1_mean	f1_std	roc_auc_mean	roc_auc_std
KNN + SMOTE	0.897223	0.001918	0.191568	0.028059	0.654010	0.031380	0.295390	0.033560	0.841936	0.008959
Support Vector Classification + SMOTE	0.802568	0.005229	0.129372	0.015525	0.868309	0.012156	0.224738	0.022536	0.908493	0.008309
Random Forest + SMOTE	0.968516	0.005635	0.520614	0.078840	0.737689	0.041818	0.607401	0.060996	0.976041	0.008811
Balanced Random Forest Classifier + SMOTE	0.970421	0.003775	0.536691	0.066390	0.760858	0.041504	0.627676	0.054254	0.977785	0.007789
Naive Bayes + SMOTE	0.832749	0.008652	0.140570	0.018413	0.791908	0.035884	0.237904	0.024481	0.878570	0.014675
KNN + Random Over Sampler	0.934925	0.002453	0.257292	0.046022	0.506129	0.020327	0.338802	0.037715	0.768037	0.010978
Support Vector Classification + Random Over Sampler	0.804172	0.006549	0.130412	0.016383	0.868309	0.012156	0.226267	0.023738	0.909158	0.008161
Random Forest + Random Over Sampler	0.983655	0.001700	0.820387	0.039207	0.645947	0.045861	0.721345	0.030247	0.976723	0.009695
Balanced Random Forest Classifier + Random Over Sampler	0.983956	0.002927	0.818265	0.043077	0.665154	0.055942	0.732100	0.038774	0.979753	0.007334
Naive Bayes + Random Over Sampler	0.848392	0.008708	0.156986	0.023001	0.816251	0.038388	0.262108	0.029229	0.907720	0.010537
KNN + Oversample using Adaptive Synthetic	0.891106	0.003488	0.185201	0.027947	0.674532	0.052608	0.289801	0.036171	0.841798	0.016659
Support Vector Classification + Oversample using Adaptive Synthetic	0.783416	0.006228	0.120807	0.017802	0.881904	0.011441	0.212025	0.026901	0.911023	0.007192
Random Forest + Oversample using Adaptive Synthetic	0.968816	0.005502	0.525135	0.083754	0.755456	0.035710	0.615613	0.059026	0.977906	0.006176
Balanced Random Forest Classifier + Oversample using Adaptive Synthetic	0.968716	0.005401	0.524354	0.084785	0.750387	0.031514	0.612917	0.057790	0.979561	0.005075
Naive Bayes + Oversample using Adaptive Synthetic	0.797052	0.008090	0.125462	0.018541	0.857940	0.018907	0.218365	0.027485	0.895868	0.018421
KNN + SMOTE-Tomek Links	0.969317	0.003625	0.603630	0.075529	0.236736	0.035188	0.335613	0.029848	0.774400	0.012179
Support Vector Classification + SMOTE-Tomek Links	0.968213	0.004694	0.920000	0.097980	0.046872	0.016269	0.088360	0.028998	0.903975	0.015879
Random Forest + SMOTE-Tomek Links	0.984758	0.002902	0.872041	0.051080	0.637534	0.047000	0.734660	0.033399	0.981178	0.007483
Balanced Random Forest Classifier + SMOTE-Tomek Links	0.893213	0.011248	0.234569	0.040648	0.971700	0.016487	0.376267	0.051417	0.980837	0.002493
Naive Bayes + SMOTE-Tomek Links	0.959490	0.004583	0.341037	0.083926	0.219317	0.049370	0.261295	0.048549	0.907392	0.009169

Figura 11: Desempenho de varios modelos com varios Oversamplers

Com base nos resultados obtidos podemos concluir que:

- O melhor modelo é o Balanced Random Forest Classifier juntamente com o SMOTE-Tomek Links, o qual obteu um auc de 98% e um f1-score de 73%, e quase em igualdade o Balanced Random forest juntamente com o SMOTE, com uma accuracy de 97% e um auc de 97%.

- O modelo de OverSampling que teve piores resultados foi o Adaptative Synthetic, juntamente com os modelos do KNN que teve um pior e o Naive Bayes

Com base nesta analise nas próximas secções iremos usar os modelos que melhor se comportaram para treinar o dataset e apresentar os seus resultados.

5.1 Suport Vector Classification + SMOTE

Aqui analisamos o desempenho do modelo Support Vector Machine(SVM) em conjunto com a técnica de oversampling SMOTE. Ao treinar o modelo obtivemos os seguintes resultados:

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.90	0.94	1910
1	0.29	0.94	0.44	84
accuracy			0.90	1994
macro avg	0.64	0.92	0.69	1994
weighted avg	0.97	0.90	0.92	1994

Figura 12: SVM - Classification Report

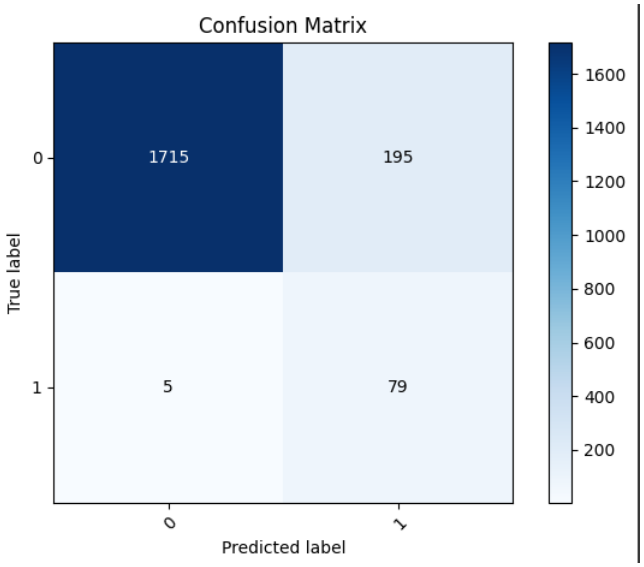


Figura 13: SVM - Confusion Matrix

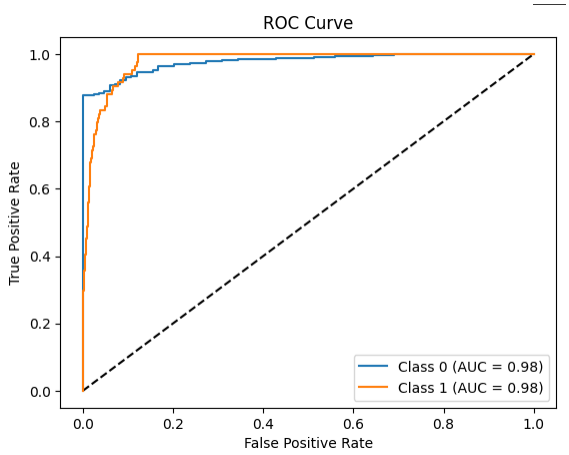


Figura 14: SVM - ROC Curve

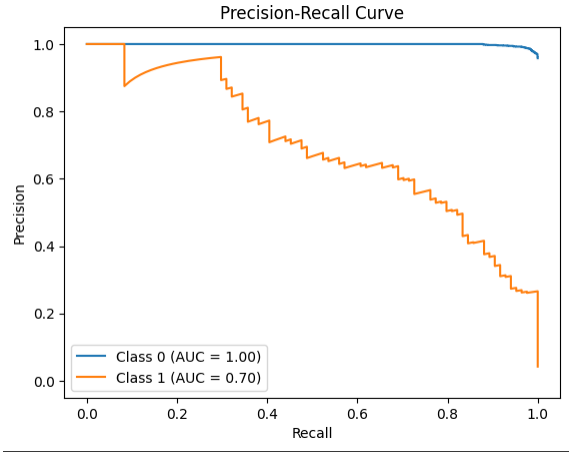


Figura 15: SVM - PCR Curve

O modelo apresentou um desempenho geral positivo, com um AUC de 98%, indicando uma boa capacidade de distinguir entre valores positivos e falsos. No entanto, houve uma diferença significativa no F1-score entre as duas classes de saída. O modelo teve um desempenho muito bom na classificação da saída 0, mas ainda precisa de melhorias na classificação da saída 1. Destaca-se a importância do pré-processamento de dados em datasets desequilibrados, como o utilizado neste estudo. É fundamental aplicar técnicas adequadas para lidar com o desbalanceamento das classes, neste caso aplicamos a técnica de oversampling SMOTE.

5.2 Balanced Random Forest Classifier + SMOTE Tomek Links

Nesta secção utilizamos o Balanced Random Forest Classifier em combinação com o oversampler SMOTE Tomek Links, e estes foram os resultados obtidos.

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.91	0.95	1910
1	0.33	0.98	0.49	84
accuracy			0.91	1994
macro avg	0.66	0.94	0.72	1994
weighted avg	0.97	0.91	0.93	1994

Figura 16: BRFC - Classification Report

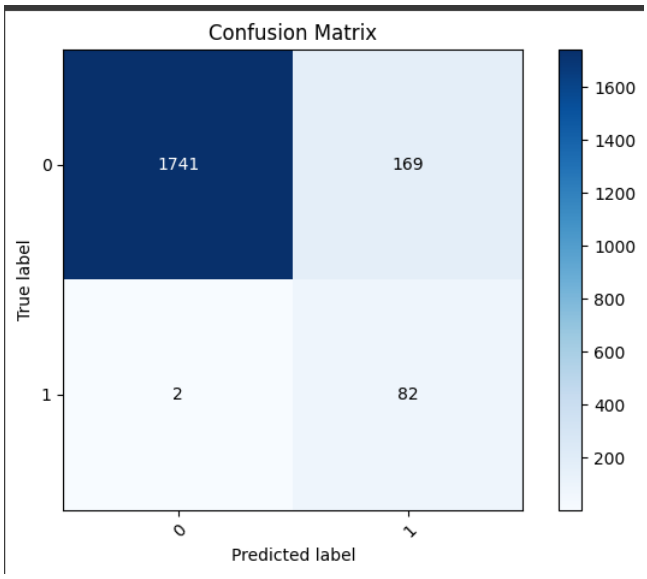


Figura 17: BRFC - Confusion Matrix

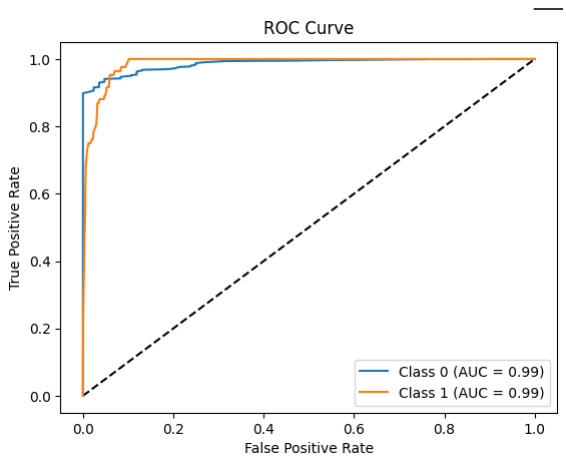


Figura 18: BRFC - ROC Curve

Podemos observar que o modelo apresenta um desempenho satisfatório na classificação da classe 0, com alta precisão e recall. No entanto, o modelo tem dificuldade em equilibrar a taxa de falsos positivos e negativos para a classe 1, resultando em baixa precisão e um F1-score mais baixo. A matriz de confusão reforça a presença de um número significativo de falsos positivos para a classe 0 .

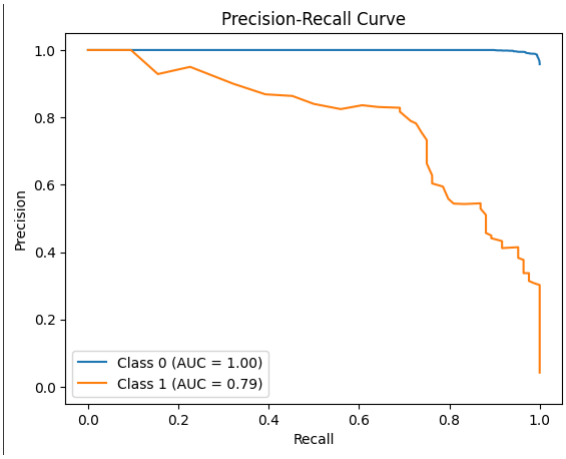


Figura 19: BRFC - PCR Curve

6 Cenario B- MulticlassClassification classification

A segunda tarefa deste projeto, que consiste em prever não apenas se ocorrerá uma falha, mas também o tipo de falha que ocorrerá definindo assim o target como *Failure Type*. Os failures type podem ser do tipo: "Power Failure", "Tool Wear Failure", "Overstrain Failure"e "Heat Dissipation Failure", de modo a treinar os modelos codificamos estes diferentes tipo de erros de forma nominal.

- Power Failure -> 1
- Tool Wear Failure -> 2
- Overstrain Failure -> 3
- Heat Dissipation Failure -> 4

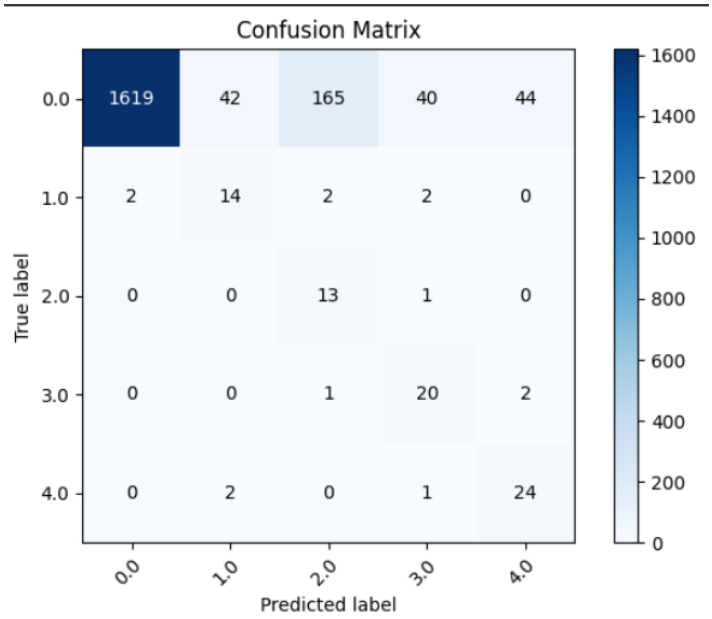
Com base nos resultados obtidos a partir do nosso modelo de otimização dos algoritmos de aprendizagem computacional e oversampling utilizado, identificamos os modelos que alcançaram os melhores resultados para a classificação binária. Agora, iremos também utilizar esses modelos para o cenário de classificação multiclasse, buscando melhorar nosso desempenho.

6.1 Balanced Random Forest + SMOTE Tomek Links

Primeiramente utilizamos o *BalancedRandomForestClassifier* que usa como classificador base o Decision Tree Classifier, e ainda utilizamos o SMOTE Tomek Links para fazer oversampling dos dados. Utilizamos este modelo com os parâmetros por Default ao que resultou nos seguintes resultados:

Classification Report:

	precision	recall	f1-score	support
0.0	1.00	0.85	0.92	1910
1.0	0.24	0.70	0.36	20
2.0	0.07	0.93	0.13	14
3.0	0.31	0.87	0.46	23
4.0	0.34	0.89	0.49	27
accuracy			0.85	1994
macro avg	0.39	0.85	0.47	1994
weighted avg	0.97	0.85	0.89	1994



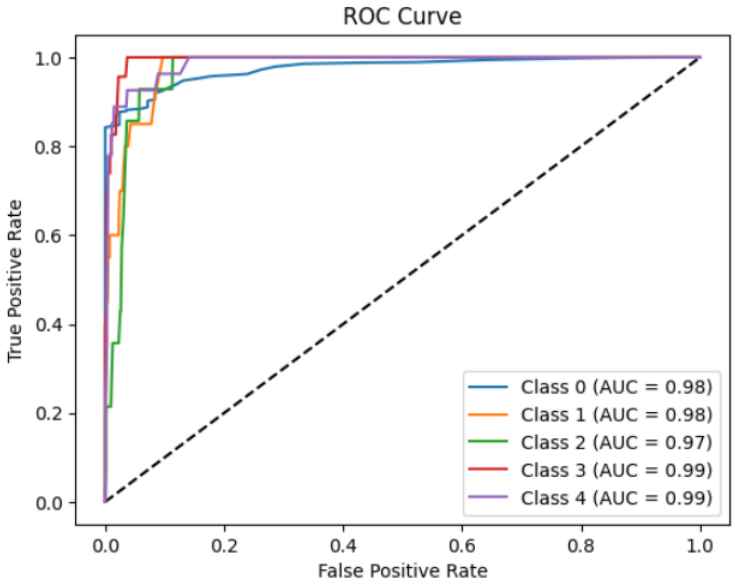
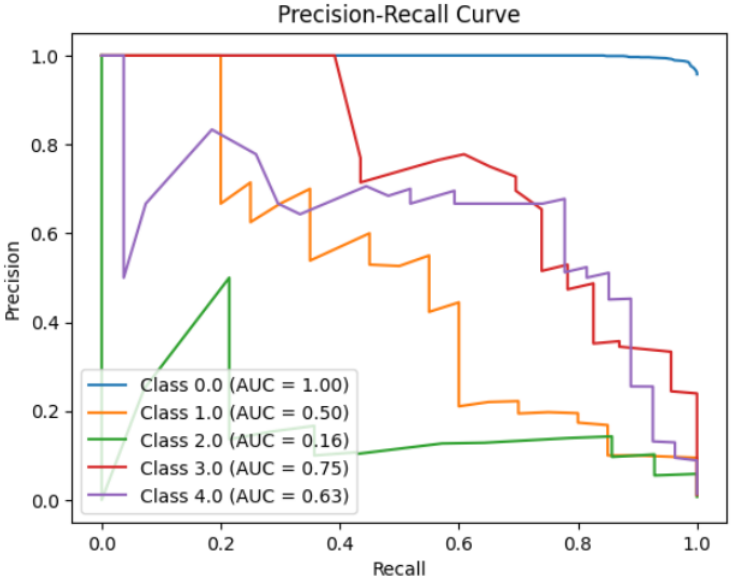


Figura 20: ROC curve - BRFD



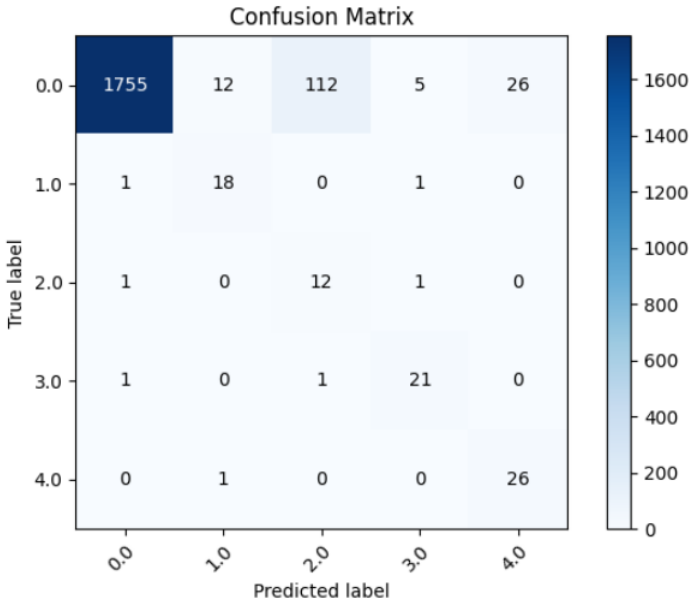
6.2 Support Vector Machine (SVC) + SMOTE

Agora avaliamos o desempenho do modelo Support Vector Machine (SVC) em combinação com a técnica de oversampling SMOTE(Synthetic Minority Over-Sampling Techinque). A seguir apresento as principais métricas utilizadas para avaliar o desempenho do modelo:

```
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      0.92      0.96      1910
         1.0       0.58      0.90      0.71        20
         2.0       0.10      0.86      0.17        14
         3.0       0.75      0.91      0.82        23
         4.0       0.50      0.96      0.66        27

    accuracy                           0.92      1994
   macro avg       0.58      0.91      0.66      1994
weighted avg       0.98      0.92      0.94      1994
```



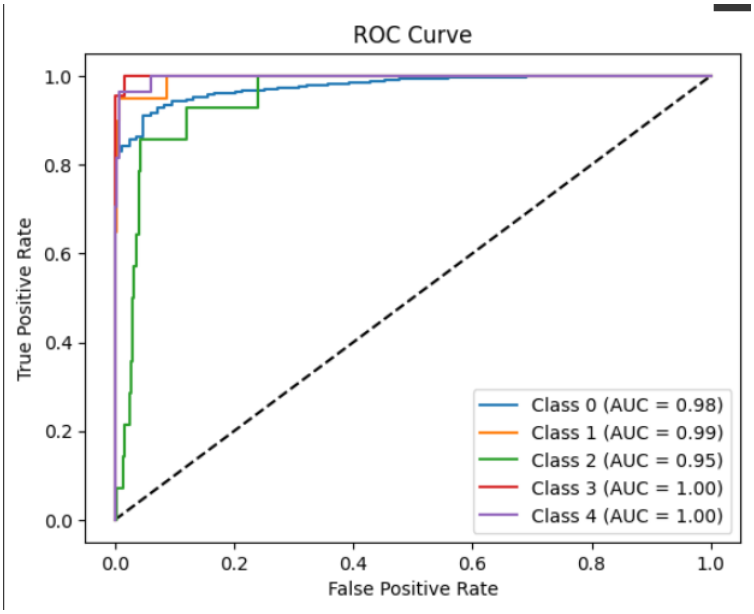
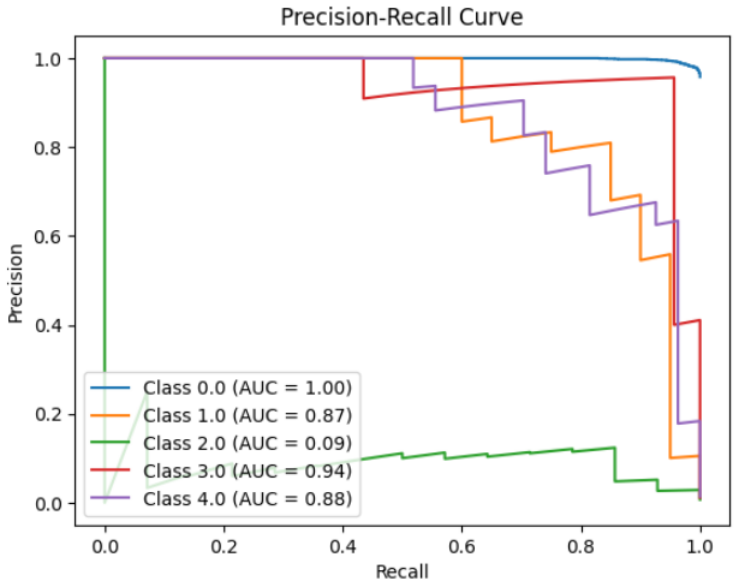


Figura 21: ROC curve - BRFD



7 Neural Networks

Para criar um modelo usando Neural Networks definimos uma função (*optimize_ mode* que realiza uma otimização do modelo da rede neuronal, utilizando a biblioteca keras e o tuner RandomSearch. O objetivo desta função é encontrar os melhores hiperparâmetros para o modelo de classificação multi-classe. A funçao vai sucessivamente adicionando as camadas do modelo, determinadas pelo hiperparâmetro *num_ layers*, e cada uma tem um número variável de neurónios este definido pelo hiperparâmetro *units_ i*.Após a definição das camadas é adicionada uma camada de normalização, *Batch Normalization*, que ajuda a estabilizar o treinamento da rede neuronal.
Por fim o modelo termina com 6 layers, e usa a função de perda *categorical_ crossentropy*, que é adequada para problemas de multiclasse e usa o otimizador *ADAM*, o qual a taxa de aprendizagem é definida e ajustada pelo hiperparâmetro *learning_ rate*.
Após a criação do modelo o tuner RandomSearch realiza uma busca aleatória para definir os hiperparâmetros. Resultando um modelo de redes neuronais otimizado.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 448)               3136

 dense_1 (Dense)             (None, 224)               100576

 dense_2 (Dense)             (None, 64)                14400

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_3 (Dense)             (None, 6)                 390

=================================================================
Total params: 118,758
Trainable params: 118,630
Non-trainable params: 128
_________________________________________________________________
```

Figura 22: Neural Network Model Summary

Depois de treinar o modelo obtivemos os seguintes resultados:
Após avaliar o modelo de rede neural proposto, podemos concluir que ele demonstrou um desempenho satisfatório no cenário de classificação multi-classe. As curvas ROC e PRC mostraram um bom poder discriminatório do modelo, com valores altos de AUC para a maioria dos targets, excepto para a classe 2. Recomenda-se explorar ajustes adicionais, como diferentes parametros e técnicas de pré-

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1936
1	0.86	0.78	0.82	23
2	0.05	0.20	0.08	5
3	0.50	0.75	0.60	8
4	0.65	0.87	0.74	23
accuracy			0.97	1995
macro avg	0.61	0.72	0.65	1995
weighted avg	0.98	0.97	0.98	1995

Figura 23: Neural Network - Classification Report

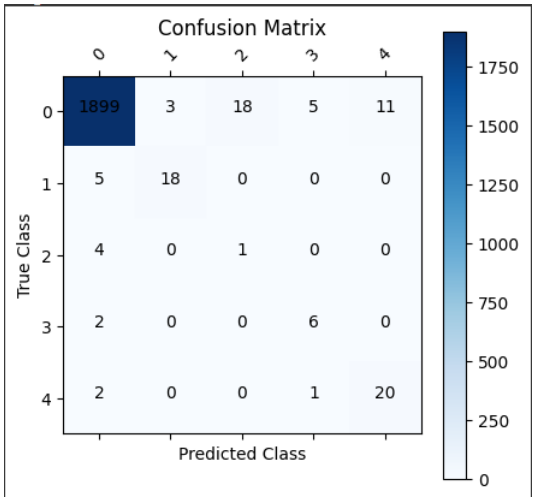


Figura 24: Neural Network - Confusion Matrix

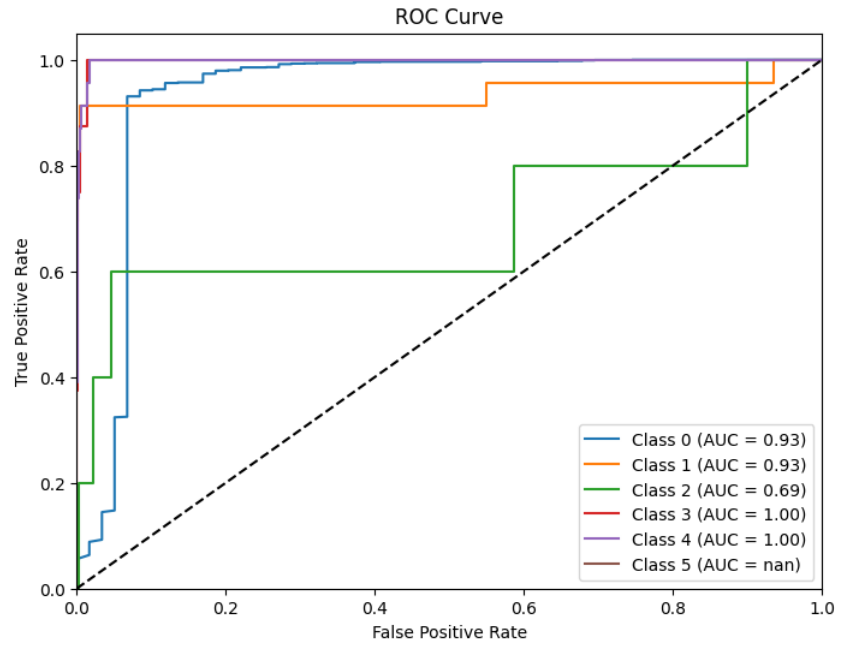


Figura 25: Neural Network - ROC Curve

processamento de dados, para melhorar o desempenho, especialmente para as classes mais desafiadoras.

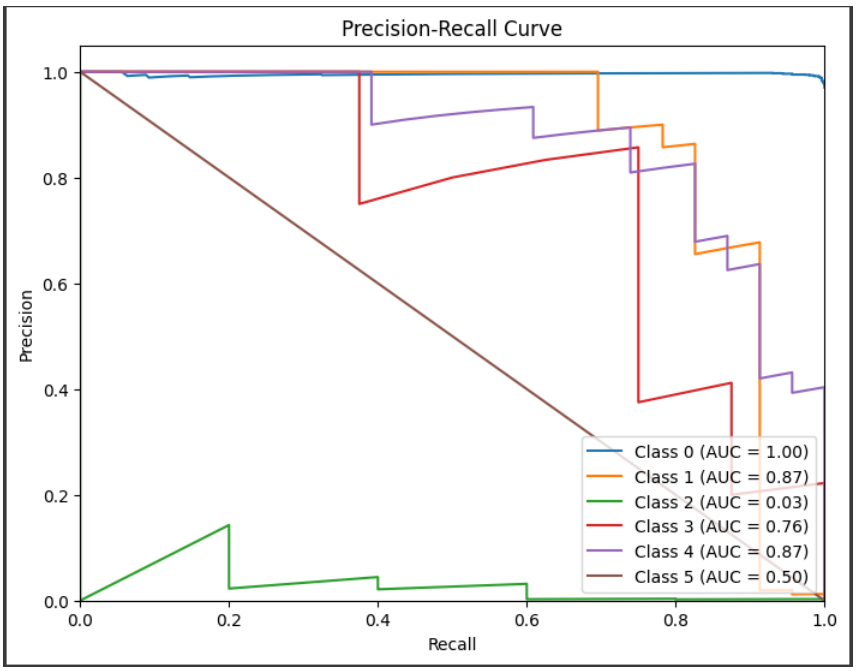


Figura 26: Neural Network - PCR

```
63/63 [==============================] - 0s 6ms/step - loss: 0.1453 - accuracy: 0.9744
[0.1453073173761367 8, 0.9744361042976379]
```

Figura 27: Neural Network - Evaluation

8 Conclusão

Em suma, algoritmos de aprendizagem computacional e as técnicas de pré-processamento utilizadas demonstraram ser eficazes na previsão de saídas tanto no cenário de previsão binário como no de previsão multi-classe. Os modelos avaliados apresentaram resultados altamente satisfatórios evidenciando a capacidade de prever as falhas das máquinas.