Projeto | Aprendizagem e Decisão Inteligentes
Grupo 2 | 2024/2025

Afonso Santos
(A104276)

João Lobo
(A104356)

Rafael Seara
(A104094)

Rita Camacho
(A104439)

Índice

1. Introdução

O presente documento apresenta informações relativas ao **Trabalho Prático** da Unidade Curricular **Aprendizagem e Decisão Inteligentes**, pertencente ao 2.º Semestre do 3.º Ano da Licenciatura em Engenharia Informática, realizada no ano letivo 2024/2025, na Universidade do Minho.

2. Metodologia

A metodologia adotada neste trabalho foi a **CRISP-DM (Cross Industry Standard Process for Data Mining)**, uma das abordagens mais reconhecidas e consolidadas para projetos de mineração de dados. A escolha desta metodologia deve-se à sua estrutura clara, flexível e iterativa, que permite uma condução sistemática e bem documentada de todas as fases do processo analítico. Além disso, a CRISP-DM é particularmente eficaz em contextos exploratórios como o presente, onde é fundamental compreender profundamente os dados antes de aplicar qualquer modelo preditivo.

A **CRISP-DM** divide-se em seis fases:

- **Compreensão do Negócio:** Definimos os objetivos do projeto, focando-nos na previsão do resultado das candidaturas.
- **Compreensão dos Dados:** Explorámos o dataset para entender os atributos, padrões e possíveis problemas (valores em falta, desequilíbrios, etc.).
- **Preparação dos Dados:** Incluiu limpeza, transformação, normalização e balanceamento dos dados.
- **Modelagem:** Testámos diferentes algoritmos (Random Forest, Decision Tree e Regressão Logística), ajustando parâmetros e técnicas auxiliares.
- **Avaliação:** Comparámos os modelos usando métricas como F1-score e Kappa para identificar o mais eficaz.
- **Implementação:** Esta fase não foi abordada, pois o foco do trabalho esteve na análise e avaliação dos modelos.

3. Datasets

3.1. Dataset Par

3.1.1. Compreensão do Negócio

O dataset "par" contém informações detalhadas sobre indivíduos que solicitaram um empréstimo bancário. Estes dados incluem diversos atributos relevantes sobre os requerentes, como o seu perfil financeiro, histórico de crédito, situação profissional, entre outros fatores que podem influenciar a decisão de aprovação, rejeição ou atraso no processamento do pedido de crédito.

Atributo Descrição

- **date** Data de nascimento do requerente.
- **person_name** Nome do requerente;
- **person_age** Idade do requerente (em anos).
- **person_gender** Género do requerente (Masculino/Feminino).
- **person_education** Nível de escolaridade mais elevado atingido pelo requerente.
- **person_income** Rendimento anual do requerente.
- **person_emp_exp** Anos de experiência profissional do requerente.
- **person_home_ownership** Situação de habitação (por exemplo: arrendada, própria, hipotecada).
- **loan_amnt** Montante do empréstimo solicitado.
- **loan_intent** Finalidade do empréstimo (ex: educação,pessoal)
- **loan_int_rate** O interesse em realizar o empréstimo.
- **loan_percent_income** Percentagem do rendimento anual que corresponde ao valor do empréstimo.
- **cb_person_cred_hist_length** Duração do histórico de crédito (em anos).
- **credit_score** Pontuação de crédito do requerente.
- **previous_loan_defaults_on_file** Indicador de existência de incumprimentos anteriores em empréstimos.
- **tax** Indicador fiscal aplicado ao requerente (pode representar a categoria ou regime fiscal).
- **loan_status** (variável alvo) Estado do empréstimo: Aprovado / Pendente / Rejeitado.

Com os atributos disponíveis neste dataset, é possível realizar uma avaliação abrangente dos dados, o que pode ser extremamente útil na identificação de padrões e condições que influenciam a decisão do banco relativamente à aprovação de um empréstimo. Por exemplo, o facto de um requerente apresentar registos de incumprimentos anteriores em empréstimos pode ser um dos principais fatores que leva à rejeição do novo pedido. Esta informação, representada pelo atributo `previous_loan_defaults_on_file`, pode indicar um histórico de risco financeiro, influenciando negativamente a confiança do banco na capacidade do indivíduo de cumprir com as suas obrigações futuras.

3.1.2. Compreensão dos Dados

Após compreendermos os significados dos atributos do dataset, analisamos o comportamento dos dados numéricos e identificamos algumas anomalias. Destacaram-se casos irrealistas, como indivíduos com 144 anos ou 125 anos de experiência profissional, o que levanta dúvidas quanto à qualidade dos dados e exige tratamento adequado, como remoção de outliers. Verificámos ainda que o atributo tax apresenta sempre o mesmo

valor, sem variação nem missing values, sugerindo pouca utilidade analítica. Por fim, a maioria dos atributos apresenta cerca de 300 valores em falta, o que requer uma abordagem cuidadosa, como imputação ou remoção, para evitar distorções nas análises e nos modelos preditivos.

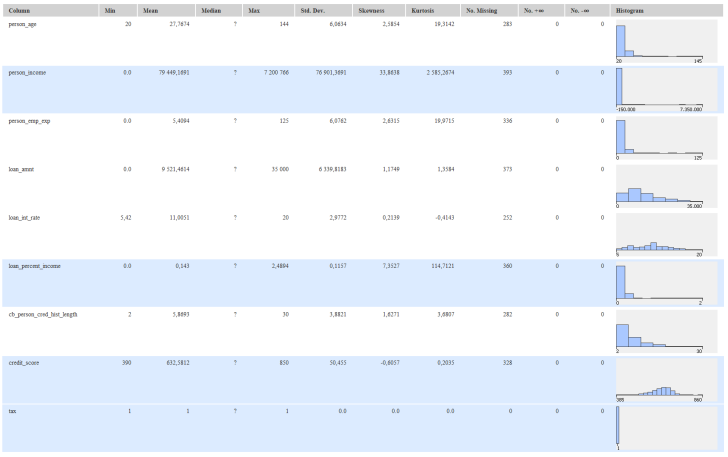


Figura 1: Visão do nodo de "Statistics - Numérico"

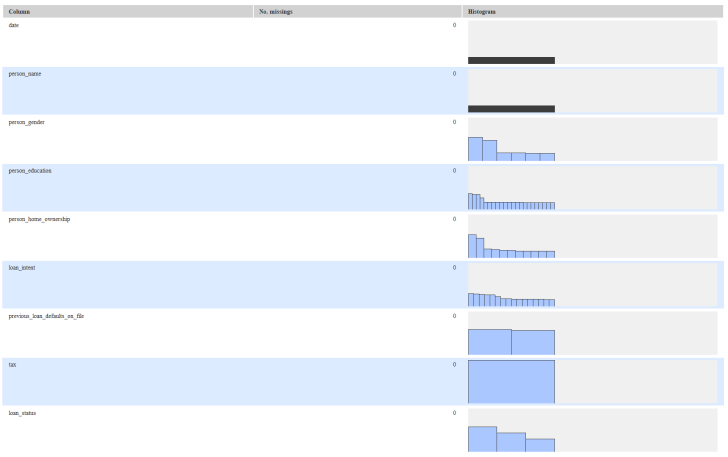


Figura 2: Visão do nodo de "Statistics - Nominal

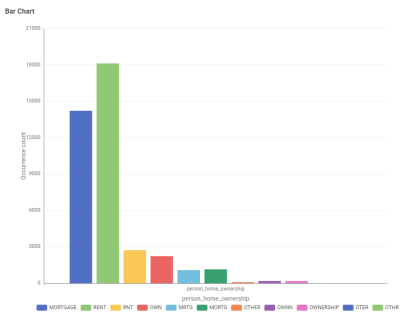


Figura 3: Bar Chart - person_home_ownership

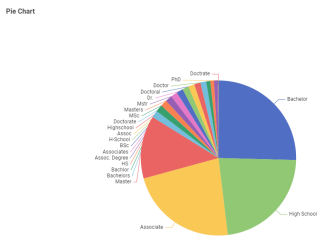


Figura 4: Pie Chart - loan_intent

Por fim, ao analisarmos a variável-alvo `loan_status`, identificamos o principal problema do dataset: o desequilíbrio acentuado entre as classes. Verificou-se que existem muitos mais casos classificados como "Rejected" e "Pending" em comparação com os casos "Approved". Este desequilíbrio pode afetar significativamente o desempenho de qualquer

modelo de classificação que se pretenda treinar, já que este tenderá a favorecer as classes mais representadas, ignorando os casos menos frequentes. Assim, será necessário aplicar técnicas de balanceamento (como **oversampling**, **undersampling** ou uso de algoritmos sensíveis ao desequilíbrio) para garantir uma análise e uma previsão mais fiáveis.

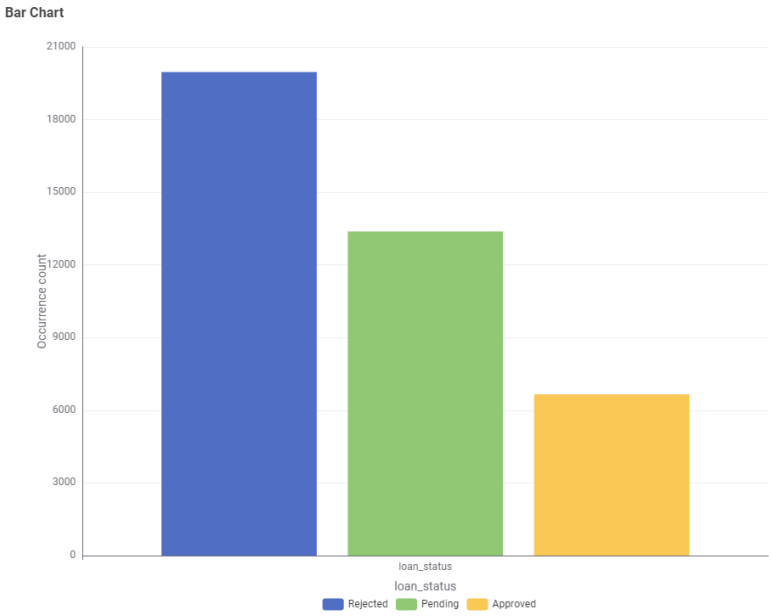


Figura 5: Bar Chart - loan_status

3.1.3. Preparação dos Dados

3.1.3.1. Normalização dos valores e Filtragem

Como referido anteriormente, vários atributos apresentavam valores inconsistentes ou mal normalizados. Assim, o primeiro passo foi uniformizar esses dados, nomeadamente nos campos relativos ao género, estado da habitação, intenção do empréstimo e nível de escolaridade, utilizando o nó **Rule Engine** para aplicar regras de normalização.

Foi também testada a técnica de criação de intervalos (bins) para a idade, com o intuito de agrupar os dados em faixas etárias. No entanto, esta abordagem revelou-se pouco eficaz, já que o número de bins gerava perda de informação ou não acrescentava valor analítico significativo.

Durante a análise, concluímos que nem o campo `person_name` nem o atributo `tax` apresentavam relevância para a nossa análise. Por isso, optamos pela sua remoção do dataset.

Relativamente à coluna date, desagregamos a informação (como o mês e o dia da semana) com o objetivo de identificar possíveis padrões temporais relevantes, como uma maior taxa de aprovação de empréstimos em determinados períodos. No entanto, não foram encontradas relações significativas.

Ainda assim, optamos por remover todos os pedidos submetidos ao fim de semana, partindo do pressuposto de que, nesses dias, os bancos estão geralmente encerrados e, portanto, os pedidos não seriam processados.

Adicionalmente, decidimos ignorar algumas linhas em que os pedidos de empréstimo foram aprovados, apesar de os clientes não apresentarem qualquer fonte de rendimento. Consideramos estes casos como erros, uma vez que, na realidade, situações deste género dificilmente resultariam na aprovação de um empréstimo.

3.1.3.2. Tratamento de Missing values

O tratamento de valores em falta foi feito de forma iterativa e progressiva, combinando diferentes estratégias e avaliando o impacto de cada uma nos resultados dos modelos.

Inicialmente, aplicamos técnicas simples como a média ou a interpolação linear para imputar valores ausentes. No entanto, ao analisarmos melhor os dados, identificamos que alguns atributos estavam relacionados matematicamente. Em particular:

loan_percent_income= loan_amnt / person_income,

Com base nesta fórmula, sempre que um destes valores estivesse ausente e os outros dois estivessem disponíveis, recalculamos o valor em falta, garantindo maior consistência e fiabilidade nos dados. O mesmo raciocínio foi aplicado à idade, quando possível.

Após os primeiros testes com modelos de machine learning, decidimos experimentar uma abordagem mais avançada, recorrendo ao **Random Forest Regressor** para imputação. Este algoritmo permite modelar relações não lineares complexas entre variáveis e usar múltiplos atributos como base para prever os valores ausentes. A sua robustez face a outliers e o menor risco de overfitting tornam-no ideal para este tipo de tarefa. O modelo foi treinado apenas com registos completos e utilizado para prever os valores em falta, com resultados satisfatórios.

3.1.3.3. Tratamento de Outliers

O tratamento de valores extremos (outliers) também foi feito de forma iterativa. Inicialmente, utilizamos um nó automático para detetar e corrigir outliers. Contudo, posteriormente desenvolvemos o nosso próprio método, através do nó **Row Filter**, que nos permitiu definir limites personalizados e ajustados ao contexto.

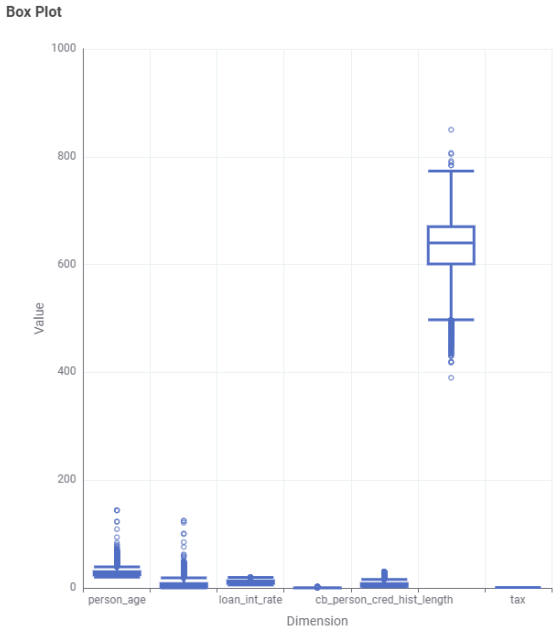


Figura 6: Antes do tratamento

Por exemplo, ao investigar as condições reais para concessão de crédito em Portugal, verificamos que a idade máxima típica para aprovação de empréstimos ronda os 75 anos. Por isso, definimos um limite superior de 85 anos, aplicando uma margem de tolerância de 10 anos. Outros outliers também foram removidos com base no seu carácter evidentemente irreal ou excessivamente divergente.

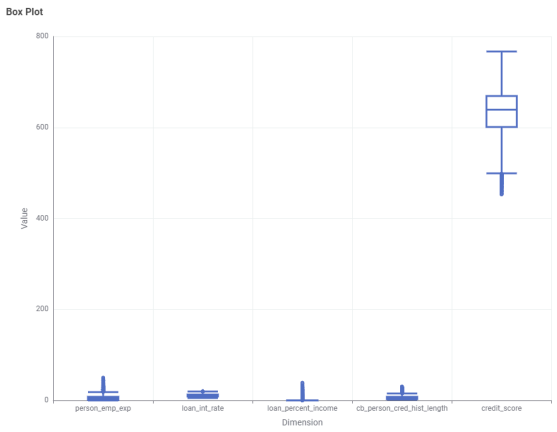


Figura 7: Depois

Concluímos que o nosso método manual com **Row Filter** personalizado produziu melhores resultados do que os métodos automáticos.

3.1.3.4. Criação de novos dados

Para melhorar a qualidade da informação e aumentar a capacidade preditiva dos modelos, foram criadas variáveis derivadas com base nas originais. Estas novas métricas procuram capturar relações implícitas entre atributos e detetar padrões relevantes. Também foram úteis para avaliar o impacto de diferentes estratégias de imputação.

As principais variáveis criadas incluem:

- **income_per_year_of_experience:** mede o rendimento por ano de experiência profissional, ajudando a identificar inconsistências.
- **loan_to_credit:** relaciona o valor do empréstimo com o crédito atribuído, sinalizando possíveis riscos.
- **loan_int_rate_to_credit_score:** avalia a adequação entre taxa de juro e score de crédito.
- **emp_exp_to_age_ratio:** indica a proporção da vida ativa, útil para detetar outliers.
- **loan_amnt_x_int_rate:** aproxima o custo total do empréstimo.
- **credit_score_x_income:** representa a força económica do requerente.

Estas transformações contribuíram para enriquecer os dados e apoiar uma melhor aprendizagem dos modelos.

3.1.3.5. Seleção de colunas

Após os tratamentos iniciais dos dados, decidimos seguir dois caminhos distintos para lidar com os valores em falta e preparar os dados para modelação:

Primeira abordagem: utilizamos o nó **Missing Value** com a técnica de média móvel (Moving Average), configurada com uma janela de 200 valores em ambos os sentidos (look behind e look ahead). Esta abordagem mostrou-se útil para suavizar o ruído presente nos dados, sobretudo em variáveis de natureza temporal.

Segunda abordagem: aplicamos um modelo de **Random Forest Regression** para estimar os valores em falta, utilizando as restantes variáveis como input. Esta técnica permite capturar relações não-lineares entre as variáveis e mostrou bons resultados em termos de imputação.

Para avaliar as correlações entre as variáveis e identificar as mais relevantes para o modelo, recorremos a duas métricas estatísticas:

- **Linear Correlation**
- **Rank Correlation**

Estas métricas permitiram identificar variáveis com um índice de correlação elevado (próximo de 1 ou -1) relativamente à variável alvo, o que sugere uma forte relação linear ou ordinal com o valor que pretendemos prever.

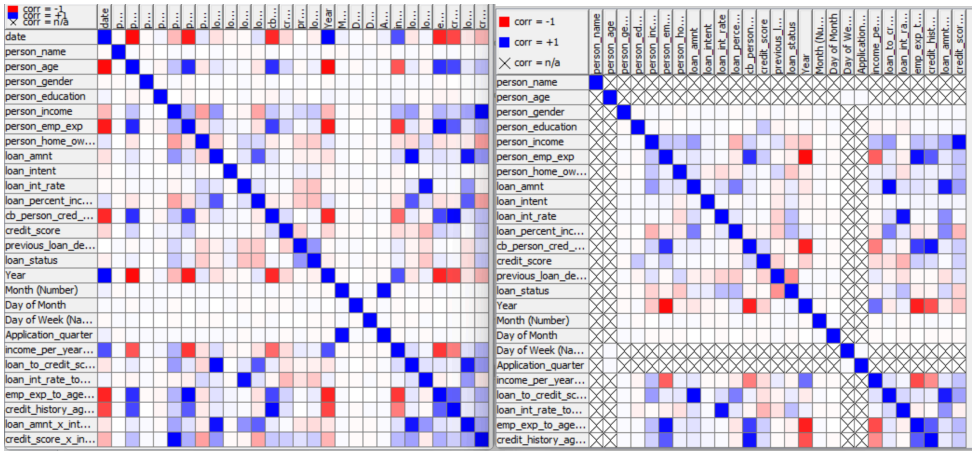


Figura 8: Rank correlion & Linear correlion com o 1 tratamento

Por outro lado, desenvolvemos e implementamos um MetaNode dedicado à codificação ordinal, que se revelou útil em alguns modelos. Esta técnica consiste em converter variáveis categóricas com hierarquia explícita (como níveis de qualidade ou satisfação) em valores numéricos ordenados, preservando a sua estrutura semântica.

Esta transformação demonstrou ser vantajosa para alguns algoritmos de `machine learning`, ao permitir que a hierarquia implícita nas categorias fosse interpretada corretamente, contribuindo assim para uma modelação mais eficaz, coerente e informativa.

Com base na análise exploratória efetuada — apoiada pelas imagens anteriormente apresentadas — e numa avaliação mais aprofundada dos dados, procedemos à seleção das variáveis mais relevantes, conforme ilustrado na imagem final desta secção.

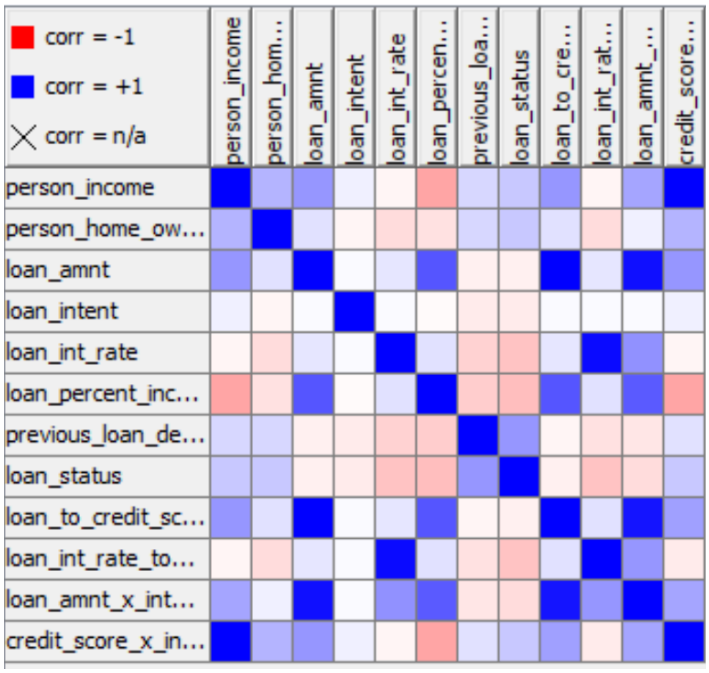


Figura 9: Colunas escolhidas

3.1.4. Avaliação de Modelagem

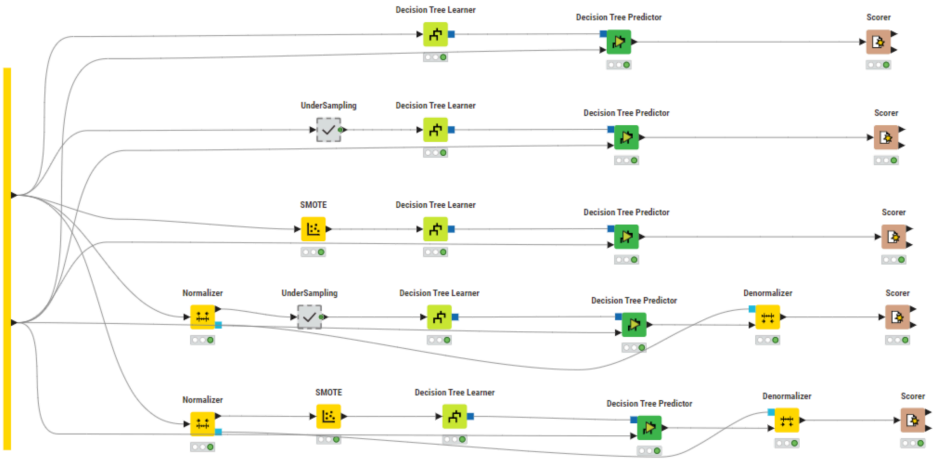
É importante realçar que, durante esta fase de desenvolvimento, foram aplicadas diversas estratégias específicas para cada modelo, com o objetivo de comparar os resultados de forma mais rigorosa e consistente. Recorremos a técnicas como **Normalizer**, **SMOTE** e um **MetaNode de undersampling**, já referido anteriormente, para equilibrar as classes e otimizar o desempenho dos modelos.

Importa ainda referir que nem todos os modelos requerem normalização dos dados. Por exemplo, modelos baseados em árvores não são sensíveis à escala das variáveis, pelo que a aplicação do normalizador nestes casos não é necessária antes da sua execução.

Estas estratégias serão descritas com maior detalhe na secção dedicada à modelação. Importa ainda referir que foi utilizada a **mesma seed** em todos os modelos, garantindo assim consistência e reprodutibilidade nos resultados obtidos.

3.1.4.1. Decision Tree

A **Decision Tree** é um dos modelos mais simples e frequentemente utilizados na criação de modelos de classificação. Baseia-se em decisões hierárquicas que dividem o conjunto de dados em subgrupos de forma intuitiva, ao longo do processo de aprendizagem. Com base nas características dos dados, o modelo separa as instâncias em diferentes categorias de saída.



Durante o desenvolvimento, foram testadas várias configurações do modelo, incluindo o ajuste do critério de divisão para **Gain Ratio**, que se mostrou mais eficaz na separação dos dados. Também foi aplicado o **pruning (poda),** com o objetivo de evitar **overfitting** e melhorar a capacidade de generalização do modelo. Além disso, aumentamos o número de threads para acelerar o processamento.

No geral, estas alterações resultaram em melhorias no desempenho, pelo que optamos por manter os ajustes aplicados na versão final do modelo.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	60.87%	0.319%
Under Sampling	51,58%	0.247%
Smote	55.24	0.269%
Normalizer com Under Sampling	35.15%	0.134%
Normalizer com Smote	35.15%	0.134%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	60.67%	0.314%
Under Sampling	50,78%	0.236%
Smote	56.62%	0.283%
Normalizer com Under Sampling	35.17%	0.134%
Normalizer com Smote	32.27%	−0.001%

3.1.4.2. Random Forest

O **Random Forest** é um modelo de ensemble baseado em múltiplas árvores de decisão, sendo amplamente utilizado pela sua robustez e capacidade de generalização. Este algoritmo constrói várias árvores em paralelo, combinando os seus resultados para melhorar a precisão e reduzir o risco de **overfitting**.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	62.13%	0.331%
Under Sampling	57,62%	0.308%
Smote	60.98%	0.325%
Normalizer com Under Sampling	45.03%	0.093%
Normalizer com Smote	44.67%	0.139%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	61.97%	0.328%
Under Sampling	57,70%	0.312%
Smote	61.12%	0.329%
Normalizer com Under Sampling	36.84%	−0.022%
Normalizer com Smote	40.93%	0.097%

Durante a fase de testes, foram experimentadas diversas configurações, nomeadamente diferentes critérios de divisão (split criterion). Após comparação, o critério que apresentou melhores resultados foi o **Information Gain**, permitindo uma divisão mais eficiente dos dados nas árvores.

Adicionalmente, o modelo final foi construído com 1.255 árvores (nós), o que contribuiu para uma maior estabilidade e desempenho nas previsões. Com base nos bons resultados obtidos, decidimos manter esta configuração para os testes finais.

3.1.4.3. Logist Regression

A **Regressão Logística** é um modelo estatístico simples e eficaz, muito utilizado em tarefas de classificação binária. Destaca-se pela rapidez no treino, baixo custo compu-

tacional e pela sua interpretação intuitiva, já que permite analisar diretamente o peso de cada variável na decisão final. Por essas razões, é frequentemente usada como modelo de referência (baseline) em projetos de machine learning.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	52.96%	0.124%
Under Sampling	48.23%	0.162%
Smote	51.73%	0.126%
Normalizer	32.32%	0.000%
Normalizer com Under Sampling	32.32%	0.000%
Normalizer com Smote	32.32%	0.000%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	53.01%	0.126%
Under Sampling	47,47%	0.154%
Smote	61.12%	0.329%
Normalizer	49.33%	−0.005%
Normalizer com Under Sampling	49.33%	−0.005%
Normalizer com Smote	32.379%	0.00%

No nosso caso, como o objetivo era prever três possíveis categorias, testamos a **Regressão Logística** em cada uma delas individualmente. O modelo foi avaliado separadamente para prever a aceitação, o adiamento e a rejeição. A configuração que apresentou melhores resultados de desempenho — nomeadamente em métricas como F1-score e Kappa — foi quando o alvo da previsão foi a classe "reject" (rejeitado). Este resultado pode estar relacionado com uma maior separabilidade dos dados nesta classe, ou seja, por conter o maior número de exemplos disponíveis, o que facilitou o treino do modelo. Além disso, ao testarmos diferentes tipos de normalização (normalizers), verificamos que os resultados se mantinham praticamente inalterados, indicando que o modelo não era sensível a essa transformação nos dados.

3.1.4.4. Gradient Boosted Trees Leaner

O **Gradient Boosted Trees (GBT)** é um modelo de ensemble baseado em árvores de decisão treinadas de forma sequencial. Ao contrário do Random Forest, que constrói as árvores em paralelo, o GBT constrói cada árvore com base nos erros da anterior, otimizando gradualmente o desempenho do modelo. Esta abordagem torna-o particularmente eficaz na correção de erros e na obtenção de previsões mais precisas.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	56.73%	0.272%
Under Sampling	52.70%	0.261%
Smote	56.45%	0.276%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	56.91%	0.274%
Under Sampling	51.896%	0.245%
Smote	56.13%	0.272

1º Tratamento de Dados	Accuracy	Cohen's kappa
Normalizer com Under Sampling	46.70%	0.118%
Normalizer com Smote	35.93%	0.017%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Normalizer com Under Sampling	41.71%	0.153%
Normalizer com Smote	35.61%	0.093%

Durante a fase de testes, foram ajustados vários hiperparâmetros, como o número de modelos (ou árvores), a profundidade máxima e a taxa de aprendizagem. Verificamos que o aumento do número de árvores contribuiu para uma melhoria consistente nos resultados, até atingir um ponto de equilíbrio entre desempenho e tempo de execução.

3.1.4.5. Naive Bayes

O **Naive Bayes** é um modelo probabilístico simples, baseado no Teorema de Bayes e na suposição de independência entre atributos. Apesar da sua simplicidade, é bastante eficaz em tarefas de classificação, especialmente quando os dados são bem distribuídos e as variáveis não estão fortemente correlacionadas.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	58.70%	0.289%
Under Sampling	50.43%	0.237%
Smote	57.40%	0.279%
Normalizer	48.98%	0.195%
Normalizer com Under Sampling	38.949%	0.138%
Normalizer com Smote	48.42%	0.183%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	58.40%	0.289%
Under Sampling	54.238%	0.272%
Smote	56.32%	0.276%
Normalizer	49.81%	0.173%
Normalizer com Under Sampling	46.58%	0.195%
Normalizer com Smote	50.335%	0.163%

Durante os testes, o modelo mostrou resultados razoáveis com um tempo de execução muito reduzido, sendo útil como uma referência rápida. Foram também testadas diferentes técnicas de normalização, mas estas não tiveram impacto significativo no desempenho final.

Embora não tenha superado modelos mais complexos como o **Random Forest** ou o **Gradient Boosted Trees**, o **Naive Bayes** destacou-se pela eficiência computacional e pela facilidade de implementação, sendo incluído na comparação pela sua leveza e capacidade de generalização em certos contextos.

3.1.4.6. Tree Ensemble

O **Tree Ensemble** é um modelo de combinação que junta várias árvores de decisão para produzir previsões mais robustas e estáveis. Ao integrar os resultados de múltiplas árvores, este modelo consegue reduzir o risco de **overfitting** e melhorar a capacidade de generalização.

Durante os testes, foram ajustados alguns hiperparâmetros, como o número de modelos a combinar. Verificou-se que o aumento moderado desse número teve impacto positivo nas métricas de desempenho, sem comprometer excessivamente o tempo de execução.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	60.75%	0.321%
Under Sampling	54.90%	0.273%
Smote	61.48%	0.318%
Normalizer com Under Sampling	42.60%	0.168%
Normalizer com Smote	50.43%	0.220%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	60.57%	0.318%
Under Sampling	55.95%	0.290%
Smote	58.63%	0.301%
Normalizer com Under Sampling	52.302%	0.265%
Normalizer com Smote	45.80%	0.179%

Apesar de partilhar algumas semelhanças com o **Random Forest** e o **Gradient Boosted Trees**, o **Tree Ensemble** destacou-se por apresentar um equilíbrio sólido entre precisão e eficiência. Durante os testes, o hiperparâmetro que mais contribuiu para a melhoria do desempenho foi o **Gain Ratio** como critério de divisão (gain index). Além disso, o aumento no número de modelos combinados revelou-se especialmente eficaz, sendo esta configuração a que apresentou os melhores resultados nas métricas avaliadas, justificando a sua inclusão como uma das abordagens principais na comparação final.

3.1.4.7. MLP Leaner

O **MLP Learner (Perceptrão Multi-Camada)** é um modelo de rede neuronal artificial composto por várias camadas de neurônios, sendo capaz de capturar relações não lineares entre as variáveis. É especialmente útil quando os dados apresentam padrões complexos que modelos lineares não conseguem representar adequadamente.

Durante os testes, foram ajustados alguns hiperparâmetros, como o número de camadas ocultas e de neurônios por camada. Neste caso, fixamos o número de camadas escondidas em 3, uma vez que estávamos a prever três categorias: pending, approved e rejected. Após experimentação, verificou-se que aumentar o número de neurónios por camada teve um impacto positivo no desempenho, sendo esta a configuração que apresentou os melhores resultados entre as várias testadas.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	60.42%	0.312%
Under Sampling	54.94%	0.279%
Smote	59.00%	0.295%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	60.26%	0.312%
Under Sampling	54.61%	0.28%
Smote	59.33%	0.302%

Embora este modelo requeira mais tempo de treino, com uma configuração equilibrada foi possível obter bons resultados, sobretudo em cenários onde as classes apresentavam sobreposição. Adicionalmente, foi necessário aplicar a codificação ordinal (já anteriormente referida) para converter variáveis categóricas em valores inteiros, assegurando a correta interpretação dos dados pelo modelo.

Apesar de ser mais sensível à normalização dos dados, o **MLP Learner** demonstrou um desempenho competitivo, sendo considerado uma abordagem complementar valiosa no conjunto de modelos testados.

3.1.4.8. PNN Leaner

O **PNN Learner (Probabilistic Neural Network)** é um modelo baseado em redes neurais que utiliza estimativas de densidade probabilística para realizar a classificação. A sua principal vantagem é a elevada rapidez na fase de treino e a capacidade de adaptação a diferentes distribuições de dados.

Durante os testes, o modelo foi avaliado com diferentes configurações de normalização e ajustes nos parâmetros de suavização. Verificou-se que o desempenho do PNN é fortemente influenciado pela forma como os dados são pré-processados, sendo a normalização um fator crítico.

1º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	51.96%	0.289
Under Sampling	54.68%	0.286%
Smote	53.90%	0.219%

2º Tratamento de Dados	Accuracy	Cohen's kappa
Só o modelo	57.96%	0.289%
Under Sampling	54.68%	0.286%
Smote	53.90%	0.219%

Apesar de ser menos flexível do que modelos como o MLP ou o Random Forest, o PNN Learner apresentou bons resultados em cenários com classes bem separadas, mostrando-se útil como alternativa leve e eficiente para determinadas tarefas de classificação.

3.1.5. Conclusões e anotações

Após uma revisão minuciosa dos resultados do treino de dados, ficamos surpreendidos ao verificar que o primeiro tratamento de dados, baseado na correção de valores através de equações matemáticas entre colunas com relações conhecidas e na média

móvel, apresentou melhores desempenhos em vários modelos, quando comparado com o segundo tratamento, baseado na **regressão com Random Forest**.

Isto foi inesperado, uma vez que a utilização de médias é, geralmente, considerada uma abordagem **simplista** e menos eficaz na correção de dados. Assumíamos que a **regressão com Random Forest** produziria valores mais fiéis à realidade, devido à sua capacidade preditiva. No entanto, em grande parte dos modelos testados, o primeiro tratamento revelou-se mais eficaz. Ainda assim, o segundo tratamento demonstrou ser pertinente em alguns modelos específicos, como no **PNN Learner** e **na regressão logística**, onde permitiu ganhos de desempenho consideráveis.

Além disso, analisamos os efeitos da técnica de **undersampling** para corrigir o desbalanceamento das classes. Concluímos que esta técnica foi prejudicial, uma vez que resultou numa perda significativa de informação útil. Fizemos um teste adicional onde reduzimos o desbalanceamento sem o eliminar completamente, e os resultados mostraram um maior desempenho dos modelos em quase todos os modelos, sugerindo que a remoção excessiva de dados (como feito inicialmente) comprometeu a qualidade do treino.

1º Tratamento de Dados	Accuracy (com desbalanceamento)	Accuracy (sem desbalanceamento)
Decision Tree	50.39%	51.58%
Random Forest	59.08%	57.62%
Logist Regression	48.34%	48.23%
Gradient Boosted	51.14%	52.70%
Naive Bayes	54.05%	50.44%
Tree Ensemble	56.54%	54.50%
MLP Learner	57.18%	54.94%
PNN Learner	53.56%	54.68

Quanto ao uso do **SMOTE**, este mostrou-se por vezes benéfico, mas é importante notar que, ao gerar dados sintéticos, nem sempre conseguiu capturar a verdadeira natureza dos dados, especialmente porque a amostra da classe "Aproved" era bastante reduzida. Assim,acreditamos que os dados artificiais criados pelo **SMOTE** não representaram fielmente a realidade, o que limitou a eficácia desta abordagem.

Por fim, o modelo com melhor desempenho geral foi o **Random Forest**, sem qualquer tipo de normalização nem correção do desbalanceamento. Isto pode indicar que o modelo consegue lidar de forma robusta com os dados tal como estão, tirando partido da sua natureza ensemble e da sua capacidade de lidar com variáveis de diferentes escalas e distribuições.

3.2. Dataset Escolhido

3.2.1. Compreensão do Negócio

O dataset escolhido neste trabalho contém informações detalhadas sobre o desempenho de equipas num ambiente industrial. Estes dados incluem diversos atributos relacionados com a organização do trabalho, condições operacionais, planeamento de produção e fatores humanos que influenciam diretamente a produtividade real alcançada por cada equipa. Entre os campos registados encontram-se indicadores como a data e o dia da semana, o departamento responsável pela tarefa, a equipa atribuída, metas de produtividade, valores padrão de tempo de produção (SMV), número de trabalhadores envolvidos, tempo de inatividade e incentivos atribuídos. Este conjunto de dados fornece uma base rica para a análise e modelação da produtividade, permitindo explorar relações complexas entre variáveis de contexto e desempenho efetivo.

Segue-se uma breve descrição dos campos utilizados:

- **date:** data de registo da observação, útil para analisar tendências temporais ou sazonalidade.
- **quarter:** trimestre do ano, permitindo relacionar a produtividade com períodos do ano.
- **department:** departamento de produção responsável pela tarefa.
- **day:** dia da semana em que o registo foi feito, o que pode ajudar a detetar padrões semanais.
- **team:** identificador numérico da equipa, permitindo segmentar os dados por grupo de trabalho.
- **targeted_productivity:** produtividade alvo estabelecida para a equipa nesse dia.
- **smv (Standard Minute Value):** tempo padrão (em minutos) necessário para produzir uma unidade, refletindo a complexidade da tarefa.
- **wip (Work In Progress):** quantidade de trabalho ainda em curso, podendo indicar a pressão ou carga de trabalho existente.
- **over_time:** tempo extra realizado pela equipa além do horário normal.
- **incentive:** valor monetário atribuído como incentivo à equipa.
- **idle_time:** tempo total (em minutos) em que os trabalhadores estiveram inativos.
- **idle_men:** número de trabalhadores inativos durante o turno.
- **no_of_style_change:** número de mudanças de estilo ou modelo de produto no dia, o que pode impactar a eficiência.
- **no_of_workers:** número total de trabalhadores na equipa.
- **actual_productivity:** produtividade real alcançada pela equipa, expressa numa escala de 0 a 1 (sendo 1 a produtividade ideal).

Com os atributos disponíveis neste dataset, é possível realizar uma avaliação abrangente dos fatores que impactam a produtividade de equipas industriais. Esta análise é particularmente útil para identificar padrões e condições que contribuem para desvios entre a produtividade esperada e a real. Por exemplo, o número de mudanças de estilo ao longo do dia — representado pelo atributo no_of_style_change — pode indicar variações no processo de produção que exigem adaptações por parte da equipa, afetando

negativamente o ritmo e a eficiência. Compreender estes elementos permite não só prever com maior precisão a produtividade futura, como também apoiar a tomada de decisões estratégicas na gestão de operações.

3.2.2. Compreensão dos Dados

Para compreender melhor os dados relacionados com a produtividade em ambiente industrial, foram utilizados diversos nodos da plataforma KNIME com o objetivo de explorar e diagnosticar o comportamento dos atributos presentes no dataset.

Através do **nodo Data Explorer**, foi possível identificar a presença de **valores em falta (missing values)** no atributo **wip**, o que exige tratamento adequado antes da aplicação de modelos preditivos. Além disso, foi detetada uma anomalia categórica no atributo **quarter**, que por vezes assume o valor **quarter5**, claramente inválido, uma vez que só existem quatro trimestres num ano.

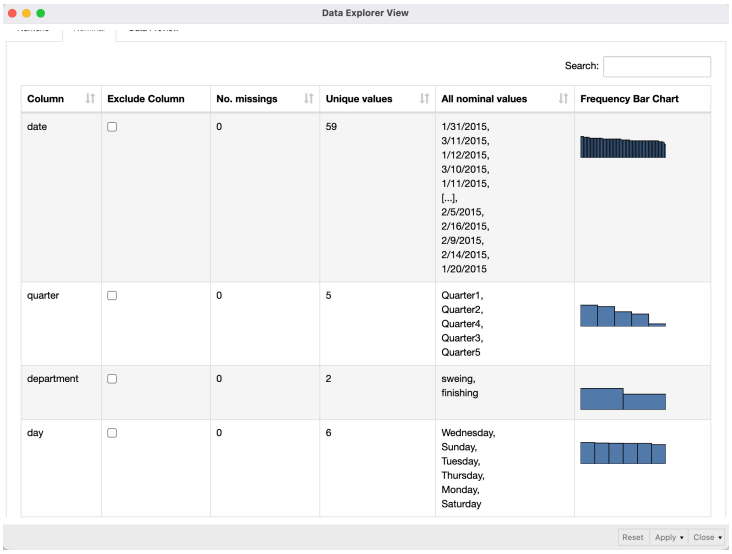


Figura 11: Visão do nodo de Data Explorer

Com o **nodo Statistics**, observámos que alguns atributos, nomeadamente **idle_time**, **idle_men** e **over_time**, apresentam valores **praticamente sempre iguais a zero**, o que indica que têm **baixa variabilidade** e **pouca utilidade analítica** para efeitos preditivos. Por outro lado, identificámos também que o atributo **actual_productivity**, que representa a produtividade real, contém **valores superiores a 1**, o que é impossível no contexto do problema e indica a presença de erros nos dados.

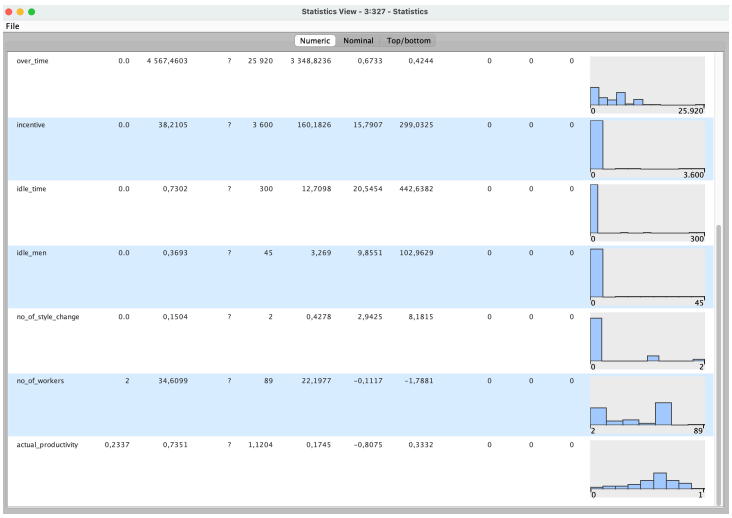


Figura 12: Visão do nodo de Statistics

Com o apoio do **nodo Rank Correlation**, verificou-se que **não existem correlações fortes** entre os atributos numéricos do dataset, o que indica que não há relações lineares evidentes entre as variáveis disponíveis.

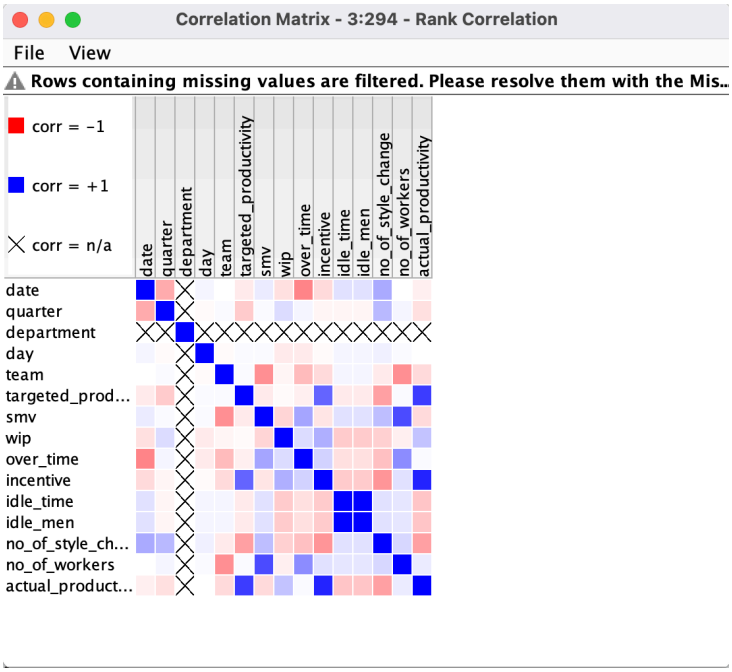


Figura 13: Visão do nodo de Rank Correlation

A utilização do **nodo Box Plot** permitiu identificar **outliers significativos** em várias variáveis, em especial no atributo **wip** e **over_time**, que revelou valores anómalos extremos, potencialmente devido a erros de inserção ou casos atípicos de produção.

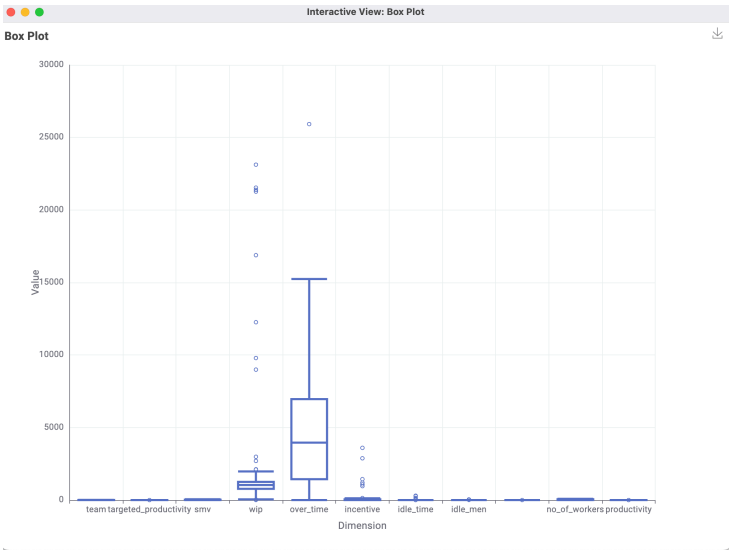


Figura 14: Visão do nodo de Box Plot

Posteriormente, recorreu-se ao **nodo Scatter Plot** para analisar a dispersão dos dados e localizar anomalias adicionais. Foram detetados casos específicos de **valores absurdos**, como por exemplo:

- O valor do over_time na **linha 146**;
- Os valores anómalos de incentive nas **linhas 1128, 1080, 960, 1440, 1200, 1130 e 1133**;
- O valor invulgar de no_of_workers na **linha 355**;
- Anomalias adicionais no wip nas **linhas 570, 572, 568, 561, 563 e 569**.

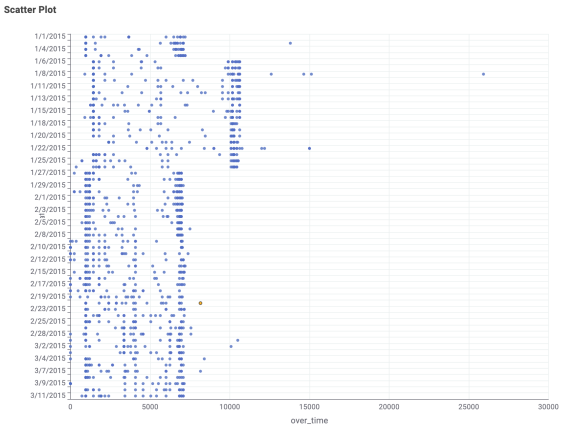


Figura 15: Scatter Plot - over_time

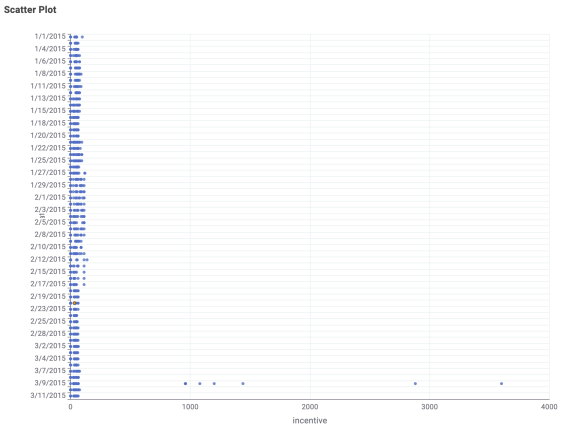


Figura 16: Scatter Plot - incentive

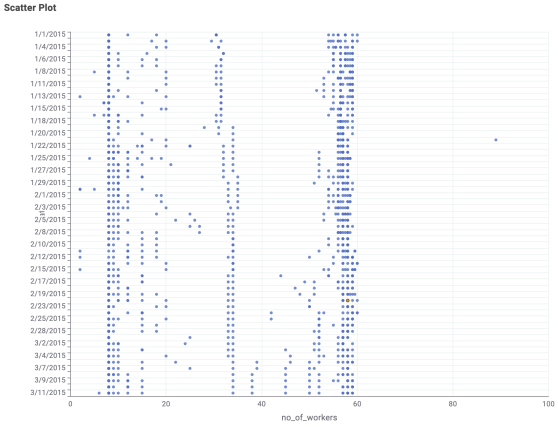


Figura 17: Scatter Plot - no_of_workers

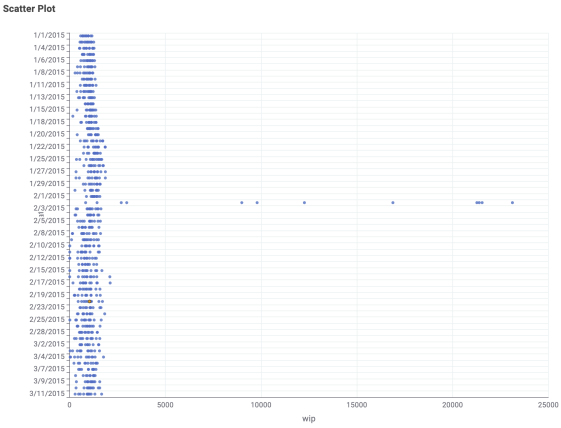


Figura 18: Scatter Plot - wip

3.2.3. Preparação dos Dados

Para dar início à preparação dos dados, foi utilizado um pipeline simples com os nodos **Linear Regression Learner** e **Regression Predictor**, com o objetivo de testar e validar diferentes estratégias de tratamento e transformação dos dados, analisando o impacto direto no desempenho preditivo através do **nodo Numeric Scorer**.

A primeira etapa de preparação consistiu na **remoção de valores inválidos** nos atributos **quarter** e **actual_productivity**. O atributo **quarter** continha registos com o valor quarter5, que não é válido no contexto de trimestres de um ano, enquanto **actual_productivity** apresentava valores superiores a 1, o que também é impossível na realidade do problema. Para proceder à remoção desses registos, foi utilizado o **nodo Row Filter**.

Posteriormente, removemos atributos considerados pouco relevantes ou redundantes. Especificamente, os atributos **idle_time**, **idle_men** e **no_of_style_change** foram eliminados com o **nodo Column Filter**, dado que apresentavam **valores praticamente sempre nulos ou invariáveis**, não contribuindo significativamente para a explicação da produtividade.

Para lidar com **outliers numéricos**, foi utilizado o **nodo Numeric Outliers**, que permitiu identificar e remover linhas com valores considerados anómalos em diferentes atributos, como **wip**, **incentive** e **over_time**. Esta medida visa reduzir o impacto de valores extremos no processo de modelação e melhorar a robustez dos resultados.

Adicionalmente, numa primeira abordagem, optámos por **remover todas as linhas com valores em falta (missing values)**, utilizando o **nodo Missing Value**, de forma a simplificar a pipeline inicial e obter um primeiro score base do modelo de regressão.

Após estas etapas de preparação, o primeiro modelo foi avaliado com o **nodo Numeric Scorer**, permitindo calcular métricas como o erro quadrático médio e o coeficiente de determinação (R^2). A partir destes resultados iniciais, foram realizados testes iterativos para explorar **alternativas de preparação de dados**, comparando diferentes abordagens de limpeza, seleção de atributos e transformação de dados.

	R^2	*MAE*	*MSR*
Tratamento inicial	0,709	0,034	0,002
Invés de retirar as linhas com o atributo ***actual_productivity*** maior que 1, substituir esses valores por 1	0,844	0,037	0,003
Eliminar apenas os outliers abaixo do limite inferior	0,843	0,037	0,003
Eliminar apenas os outliers acima do limite inferior	0,819	0,044	0,004
Substituir os missing values de ***wip*** com o valor mais frequente	0,405	0,094	0,016
Retirar coluna ***department***	0,844	0,037	0,003
Remover colunas com a correlação abaixo de 0.1 em relação à coluna ***actual_productivity***	0,841	0,037	0,003
Remover colunas com a correlação abaixo de 0.15 em relação à coluna ***actual_productivity***	0,831	0,038	0,003
Remover colunas com a correlação abaixo de 0.2 em relação à coluna ***actual_productivity***	0,831	0,038	0,003

Importa salientar que **apenas as alterações que demonstraram melhoria ou igualdade nas métricas de avaliação foram mantidas**. Caso contrário, o fluxo de trabalho regressava à versão anterior, garantindo que cada passo contribuía efetivamente para um melhor desempenho preditivo.

O melhor tratamento de dados foi, em última análise, aquele em que removemos os valores em falta, eliminámos as linhas que continham o valor **Quarter5**, substituímos os valores de produtividade superiores a 1 por 1, removemos outliers e excluímos as colunas com atributos redundantes.

3.3. Modelos e Avaliação

Começamos por utilizar Linear Regression, Gradient Boosted Trees, Random Forest Regression e Redes Neurais para a modelação de dados. Seguidamente, realizou-se a partição dos dados, recorrendo ao nodo Partitioning, dividindo 70% para treino e 30% para testes Como *seed*, utilizamos 2025 que manteve-se em todos os modelos.

Nesta secção, abordaremos os diferentes nodos de treino que utilizámos, comparando os diversos modelos, percebendo qual seria o mais adequado para o nosso problema.

Assim, as seguintes tabelas representam o trabalho realizado.

Linear Regression	R^2	MAE	MSR
Só o modelo	0.844	0.037	0.003
Modelo com Smote	0.814	0.035	0.003
Modelo com Nomarlização	0.839	0.037	0.003
Modelo com Smote e Normalização	0.816	0.035	0.002

Gradient Boosted Trees	R^2	MAE	MSR
Só o modelo	0.729	0.043	0.003
Modelo com Smote	0.922	0.033	0.003
Modelo com Nomarlização	0.844	0.052	0.003
Modelo com Smote e Normalização	0.895	0.038	0.002

Random Forest Regression	R^2	MAE	MSR
Só o modelo	0.790	0.044	0,005
Modelo com Smote	0.924	0.021	0,001
Modelo com Normalização	0.837	0.035	0,003
Modelo com Normalização e Smote	0.913	0.022	0,001

Redes Neurais	R^2	MAE	MSR
Só o modelo	0.75	0.047	0,004
Modelo com Smote	0.791	0.038	0,003

Para finalizar o processo de modelação, experimentámos abordar o problema como uma tarefa de classificação em vez de regressão, utilizando o modelo **Random Forest Regression** adaptado ao contexto classificativo. Com esta abordagem, obtivemos uma *accuracy* de **79,641%** e um valor de Cohen's kappa de 0,746.

Concluímos que o **melhor resultado** foi alcançado com o modelo **Random Forest Regression**, após a aplicação da técnica **SMOTE**, que permitiu equilibrar os dados e melhorar significativamente a performance do modelo. Os valores obtidos foram bastante positivos, com um R^2 de 0,924, um MAE de 0,021 e um MSE de apenas 0,001, indicando uma elevada capacidade preditiva e um excelente ajuste ao problema em estudo.

4. Conclusão e Melhorias

Concluindo, sentimos que nosso conhecimento de Aprendizagem e Decisão Inteligentes cresceu bastante.

Ao longo da elaboração e treino dos modelos em ambos os conjuntos de dados, adquirimos uma compreensão mais abrangente das técnicas de análise de dados, pré-processamento e construção de modelos, assim como os seus algoritmos, diferentes configurações e infinitas possibilidades alteravam o resultado final.

Reconhecemos, assim, a importância do pré-processamento de dados na melhoria da qualidade dos modelos. O tratamento de valores discrepantes, a normalização de variáveis e outras técnicas relevantes contribuíram significativamente para a precisão e generalização dos modelos desenvolvidos, melhorando a nossa visão das diversas técnicas que poderiam ser utilizadas na melhoria de modelos de aprendizagem.

Globalmente, estamos convictos de que o nosso trabalho foi bem executado e que os resultados obtidos refletem isso, apesar de admitir que há sempre espaço para melhorias.

Algumas das possíveis melhorias incluem a exploração de técnicas mais avançadas de seleção de atributos para identificar as variáveis mais impactantes, bem como criação de novas *features* a partir das já existentes nos conjuntos de dados. Estas medidas poderiam aprimorar ainda mais o desempenho dos modelos, assim como muitas outras variáveis.

5. Avaliação Pares

- Afonso Santos: 1
- João Lobo: 1
- Rafael Seara: 1
- Rita Camacho: 1