Universidade do Minho
Departamento de Informática

Aprendizagem e Decisão Inteligente

Trabalho de Grupo

Grupo 17

Realizado por:
António Santos Gil Afonso (A95180)
Gonçalo Fernandes Lemos (A96484)
Hugo Filipe Silva Abelheira (A95151)
Simão Pedro Cunha Matos (A96115)

13 de maio de 2023
Ano Letivo 2022/23

Conteúdo

1 Introdução

1.1 Contextualização

No âmbito do trabalho prático da disciplina de Aprendizagem e Decisão Inteligente, foi proposto pela equipa docente o desenvolvimento de modelos de aprendizagem através da análise e extração de conhecimento mediante certas distribuições de dados. Recorremos à plataforma *KNIME* para realizar as tarefas propostas.

1.2 Descrição do trabalho

Este projeto consiste em explorar, analisar e preparar dois *datasets*, procurando extrair conhecimento relevante no contexto dos problemas em questão.

Através da exploração de dados vamos analisar, bem como obter bastante informações sobre que tipo de tratamento de dados é que devemos fazer.

Acabando a exploração de dados, começamos então o tratamento de dados para desenvolver modelos de aprendizagem supervisionada e não-supervisionada para prever algumas variáveis dependentes e, desta forma, retirar conclusões sobre cada um dos *datasets*, mediante o respetivo contexto.

1.3 Metodologia do grupo

O nosso grupo utilizou a mesma estratégia para desenvolver o trabalho pedido para ambos os *datasets*. Começámos por fazer uma intensa exploração de dados que nos permitiu retirar algumas informações importantes, como por exemplo, como fazer a limpeza dos valores em falta, que colunas é que devem ser retiradas ou transformadas ou até definir limites de valores válidos para algumas variáveis.

Em seguida desenvolvemos a parte da visualização de dados, que é na verdade um conceito muito interdependente da exploração de dados, tanto que muitas vezes nem se faz a distinção. Esta parte é extramemente útil para reconhecer erros bem como facilita imenso a ilustração de algumas informações que não se obtém tão facilmente apenas a olhar para o conjunto de dados em "bruto". Para a realização desta fase, o nosso método foi levantar algumas questões pertinentes que nos ajudam a perceber os dados e o contexto em questão (melhor abordado na respetiva secção de cada *dataset*.

Na terceira e última fase do nosso projeto, corremos vários modelos de *Machine Learning*, como já dito anteriormente, para extrair informação e chegar a conclusões sobre os assuntos em mãos. Dividimos cada modelo no seu respetivo tratamento, na forma como fizemos a partição do modelo e na análise dos seus resultados. Temos ainda uma divisão clara entre as ténicas de aprendizagem supervisionada e não supervisionada usadas.

Neste relatório, abordaremos cada uma destas fases em separado, para cada um dos datasets, para debater e explicar tanto as nossas escolhas, como os resultados obtidos.

1.4 Descrição de cada *Dataset*

1.4.1 *Dataset* Atribuído: Produção e Vestuário

Neste *dataset* estão contidos dados que visam representar alguns dos parâmetros mais importantes para estudar a produtividade de uma linha de produção.

Como tal, o nosso objetivo principal com este *dataset* será desenvolver modelos de aprendizagem (supervisionada e não supervisionada) para prever a produtividade de cada instância.

Teremos em especial atenção modelos de regressão, visto que a única variável dependente é do tipo *double*.

No entanto, desenvolvemos vários modelos de *Machine Learning* para mostrar-mos alguns dados importantes, bem como retirar conclusões mais próximas do real e obter-mos valores mais precisos, através da comparação destes modelos.

De seguida podemos ver a lista de atributos (colunas) que constituem este conjunto de dados.

Variáveis Independentes:

- **rowID:** ID de cada entrada;
- **date:** Data de cada entrada, no formato de DD/MM/YYYY;
- **department:** Departamento associado a cada instância;
- **team:** Número de equipa associado a cada instância;
- **targeted_productivity:** Produtividade esperada de cada equipa, para um certo dia, definida pela respetiva autoridade;
- **smv:** *Standard Minute Value*, representa o tempo necessário para uma tarefa;
- **wip:** *Work in Progress*, representa a quantidade de produtos que ficaram por acabar;
- **over_time:** Representa a quantidade de tempo extra, em minutos, que cada equipa trabalhou em determinado dia;
- **incentive:** Representa o incentivo monetário (em BDT) para um certo conjunto de ações;
- **idle_time:** Quantidade de tempo que a linha de produção ficou parada;
- **idle_men:** Quantidade de trabalhadores que ficaram parados devido à interrupção da linha;
- **no_of_workers:** Número de trabalhadores por equipa;
- **no_of_style_change:** Número de mudanças no *design* de um determinado produto;

Variáveis Dependentes:

- **actual_productivity:** A produtividade que cada equipa efetivamente atingiu;

Tamanho do *dataset*: 1197 entradas;

1.4.2 *Dataset* Grupo: *Income Evaluation*

Este conjunto de dados contém informações demográficas e financeiras de indivíduos nos Estados Unidos, com o objetivo de prever o salário anual de cada instância. O nosso objetivo é, portanto, desenvolver modelos de aprendizagem supervisionada e não supervisionada para prever o valor desta variável dependente.

Modelos de classificação serão particularmente importantes, uma vez que a única variável dependente é de tipo categórica. Em contraste, será impossível correrm os modelos de regressão linear, visto que não temos forma de transformar o atributo a ser previsto numa variável contínua. Iremos ainda otimizar todos os modelos desenvolvidos com o objetivo de comparar resultados e extrair informação relevante.

Em seguida, uma descrição mais detalhada dos atributos que compõem o nosso *dataset*.

Variáveis Independentes:

- **age:** A idade do indivíduo em anos.
- **workclass:** O tipo de trabalho realizado pelo indivíduo;
- **fnlwgt:** O peso de amostragem final atribuído a cada indivíduo, que é um número estimado de quantas pessoas o indivíduo representa na população geral;
- **education:** O nível educacional do indivíduo;
- **educational-num:** O número de anos de educação completos do indivíduo;

- **marital-status:** O estado civil do indivíduo;
- **occupation:** A ocupação do indivíduo;
- **relationship:** O relacionamento do indivíduo com a família;
- **race** A raça do indivíduo;
- **gender:** O género do indivíduo;
- **capital-gain:** O ganho de capital do indivíduo em dólares americanos.
- **capital-loss:** A perda de capital do indivíduo em dólares americanos.
- **hours-per-week:** O número de horas que o indivíduo trabalha por semana.
- **native-country:** O país de origem do indivíduo.

Variáveis Dependentes:

- **income:** A variável alvo que indica se o indivíduo tem um salário anual superior ou inferior a $50.000.

Tamanho do dataset: 32561 entradas;

1.5 Notas

Alguns valores de entradas deste *dataset* são capazes de necessitar de mais explicação do que a fornecida: **Coluna: Marital-status**

- Married-AF-spouse: significa que a pessoa é casada e o cônjuge é um membro das Forças Armadas dos EUA.
- Married-civ-spouse: significa que a pessoa é casada legalmente com um cônjuge civil.
- Married-spouse-absent: significa que a pessoa é casada, mas o cônjuge não está presente.

Coluna: Workclass

- Self-emp-inc: indica que o indivíduo é proprietário de um negócio ou empresa que tem incorporação e que é responsável por sua própria renda.
- Self-emp-not-inc: indica que o indivíduo é proprietário de um negócio ou empresa que não tem incorporação e é responsável por sua própria renda.

Coluna: Education

- Assoc-adm: indica que a pessoa concluiu um programa de associado em administração de empresas
- Assoc-voc: indica que a pessoa concluiu um programa de associado em uma área profissional ou técnica específica
- Prof-school: indica que a pessoa concluiu um programa profissional em uma faculdade ou universidade, como direito ou medicina
- Some-college: indica que a pessoa frequentou uma faculdade ou universidade, mas não concluiu um curso de graduação
- HS-grad e 12th não representam a mesma coisa, 12th significa que ainda está a frequentar o último ano do liceu.

2 Exploração de Dados

A exploração de dados é uma etapa crucial em qualquer análise de dados, processo de mineração de dados ou modelagem de aprendizagem de máquina. É uma abordagem sistemática para descobrir padrões, tendências e extrair informações essenciais em grandes conjuntos de dados usando técnicas estatísticas.

A exploração de dados é a primeira etapa num processo de análise de dados e será complementada com a segunda fase do nosso projeto, a visualização de dados.

2.1 Produção e Vestuário

A exploração de dados deste *dataset*, revelou-nos imensa informação pertinente.

A primeira coisa que fizemos foi verificar os correlações entre os diferentes atributos, que acabou por ser um bocado surpreendente, visto estarmos à espera de maiores dependências.

De seguida verificamos as estatísticas do nosso modelo, que nos facilita a visualização de algumas medidas de dispersão como o máximo/mínimo, a média, o desvio padrão, entre outros. Este nodo é útil para saber que colunas normalizar ou então para identificar que colunas é que vão necessitar de tratamento de *outliers*. No caso, identificamos colunas como o *smv* e o *overtime*. É de notar que este nodo só aceita no máximo, por nossa escolha, 1000 entradas de cada vez, pelo que limitamos o conjunto de dados a esse tamanho para fins de análise.

Usamos o nodo *Data Explorer*, que do nosso ponto de vista é o nodo que nos fornece mais informação sobre o dataset. Se aplicarmos este nodo sem nenhum tratamento, ele funciona de forma muito análoga, demonstrando basicamente os mesmos parâmetros, ao nodo *statistics* para os valores numéricos. Para os valores nominais, temos parâmetros mais interessantes. Como tal, decidimos fazer a transformação de todos atributos para *String*. Este nodo é agora capaz de nos dizer a quantidade de valores em falta, a quantidade de valores únicos e ainda um excerto da lista destes valores únicos. É de notar que fizemos aqui o primeiro tratamento de *missing values*, para podermos comparar os valores de antes e depois.

Para facilitar ainda mais a visualização, incluimos um nodo *GroupBy* para podermos ver todos os valores únicos de uma ou várias colunas e um nodo *BoxPlot*, que serve para visualizar as caixas de bigodes de cada uma das colunas.

Concluíndo, através da exploração de dados, retiramos uma conclusão que nos parece ser de extrema importância ao problema. Repará-mos que a única coluna que continha *missing values* era a coluna *wip*, que, mais uma vez, representa a quantidade de produtos que necessitam de acabamento. Quando retiramos as entradas que correspondiam a estes valores em falta, reparamos que um departamente inteiro estava a ser excluído do conjunto de dados, o que nos levou à conclusão de que os valores *"NaN"* desta coluna, representam o departamento *Finishing* à espera da linha de produção do outro departamento *Sweing*.

2.2 Income Evaluation

Para este *dataset*, fizemos uma exploração de dados mais ou menos idêntica. Começamos por identificar as correlações lineares entre cada coluna, e, mais uma vez, ficamos surpreendidos pela falta de dependência entre algumas colunas, mas esta ausência é facilmente esclarecida, pois grande parte dos valores são do tipo *String*.

Durante a exploração das estatísticas, que corremos ao mesmo método de limitar o conjunto de dados a apenas 1000 entradas, é de notar que os *dataset* tem um boa distribuição de dados, com a média de colunas como a *age*, *education-num* e *hours-per-week* serem uma ótima representação daquilo que é a realidade. No entato, reparámos que o valor máximo da coluna *hours-per-week*, 99, é um valor a ter em atenção, devendo ser tratado como um *outlier* ou até mesmo retirado do conjunto de dados.

Já na exploração de dados, voltámos a fazer a análise com e sem o tratamento de missing values. Ainda assim, é de notar que neste *dataset* é necessário transformar os valores *"?"*, que fazem parte de 3 colunas diferentes, em *missing values* "válidos". Para além da quantidade de valores únicos e em falta, este nodo não nos deu muita mais informações. No entanto, ajudou-nos a escolher o tipo de tratamento que fariamos para cada coluna, algo que vai ser abordado com maior detalhe mais tarde.

De forma similar, incluimos também um nodo *BoxPlot* e um *GroupBy* para ajudar na visualização de dados. Ao contrário do *dataset: Produção e Vestuário*, o agrupamento foi especialmente importante para identificarmos um erro de interpretação. Inicialmente pensávamos que a coluna *education-num* representava apenas o número de anos que o respetivo indivíduo andava/andou na escola. Através desta análise, é identificável uma relação entre essa e a coluna *education*, como se pode verificar na seguinte imagem:

education	education-num
Preschool	1
1st-4th	2
5th-6th	3
7th-8th	4
9th	5
10th	6
11th	7
12th	8
HS-grad	9
Some-college	10
Assoc-voc	11
Assoc-acdm	12
Bachelors	13
Masters	14
Prof-school	15
Doctorate	16

Figura 1: Relação entre *education* e *education_num*

3 Visualização de Dados

A visualização de dados é outra etapa bastante importante, que complementa, e muitas vezes é agregada com, a exploração de dados.

Nesta fase, é possível que os dados sejam transformados em informações visuais, tornando mais fácil para os analistas de dados e cientistas de dados entenderem os padrões, tendências e *insights* presentes nos dados.

Desenvolvemos vários gráficos para cada um dos *dataset*, de acordo com as nossas necessidades e questões, para tentar entender melhor o âmbito do problema. Ambos os resultados desta visualização de dados foram bastante positivos, permitindo-nos esclarecer algumas dúvidas bem como corrigir algumas assunções.

Segue-se então uma explicação mais detalhada sobre algumas das mais interessantes visualizações de dados que desenvolvemos para cada conjuntos de dados:

3.1 Produção e Vestuário

Em relação ao *dataset* da fábrica, não podemos deixar de ter em conta a nossa variável dependente, *actual_productivity* que representa então o rendimento de cada equipa, em cada dia. Desta forma, decidimos então ver como é que este rendimento varia:

Pelo departamento:

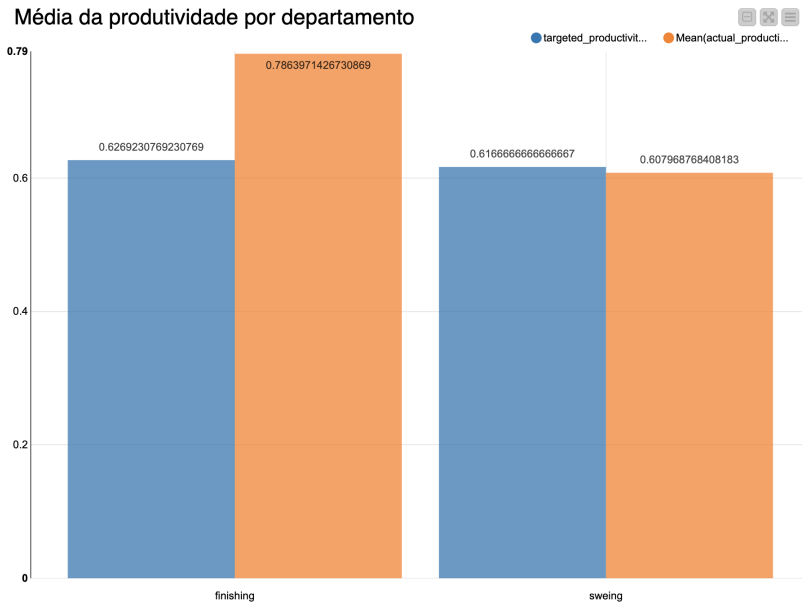


Figura 2: Média da produtividade por departamento

Por equipa (Departamento sweing:)

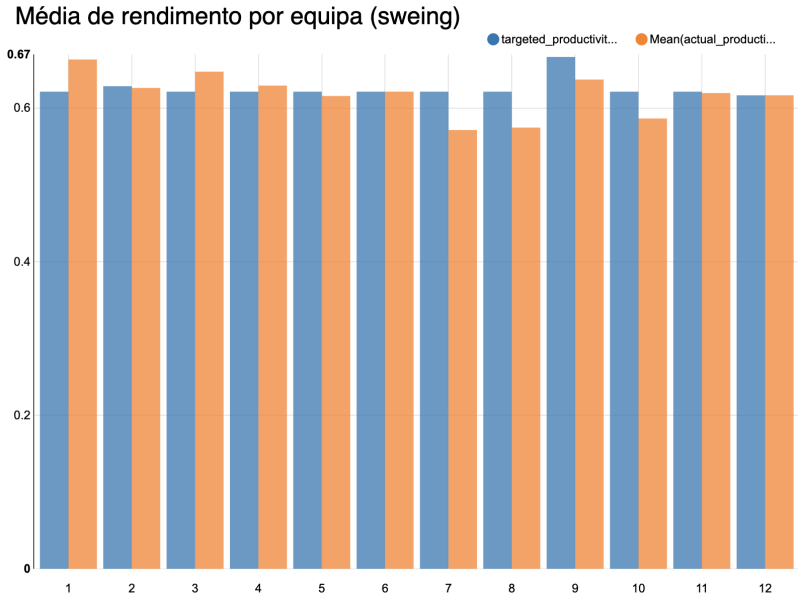


Figura 3: Média do rendimento por equipa

Por equipa (Departamento finishing:)

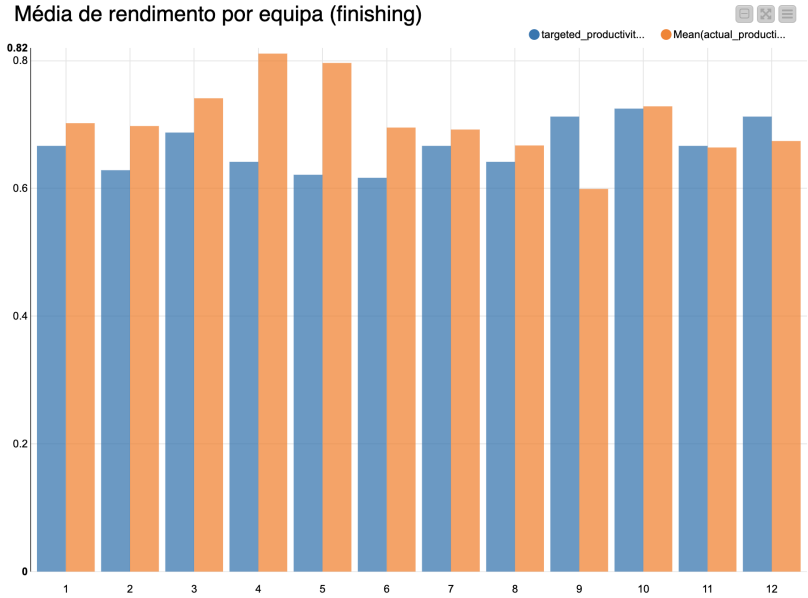


Figura 4: Média do rendimento por equipa

Através destes gráficos, podemos verificar que o departamento *sweing* não só tem uma média de rendimento menor do que o esperado, mas também fica bastante atrás do departamento *finishing*.

No entanto, às vezes a média pode ser uma métrica enganadora, e por isso decidimos desenvolver mais alguns gráficos, e percebemos dois fatores importantes que podem influenciar esta média do rendimento. Primeiro, é importante referir que o departamento *sweing* tem muitos mais trabalhadores:

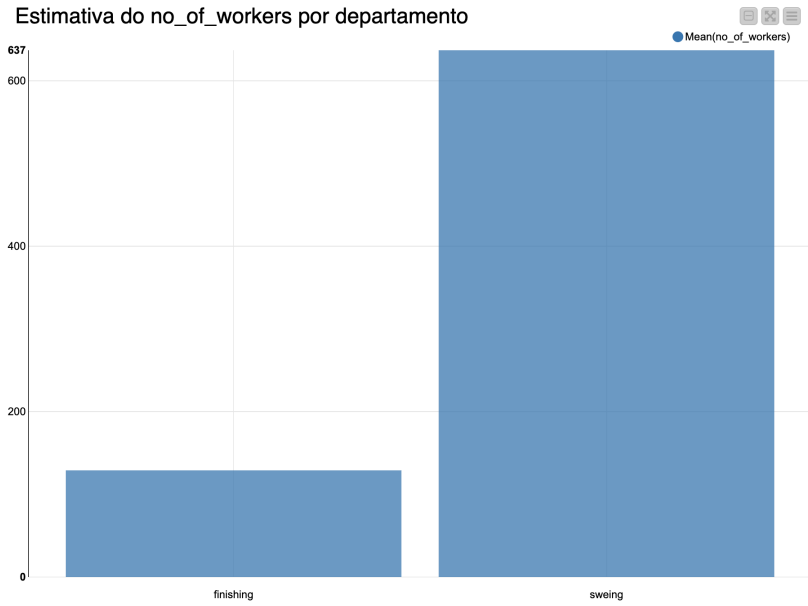


Figura 5: Estimativa do número de trabalhadores por departamento

Em segundo lugar, é de notar que o *SMV - Standard Minute Value*, que como dito anteriormente, representa o tempo necessário para realizar uma tarefa, é bastante inferior para o departamento *finishing*, o que indica que é necessário menos tempo, ou seja torna-se menos trabalhoso, para realizar uma tarefa nesta área.

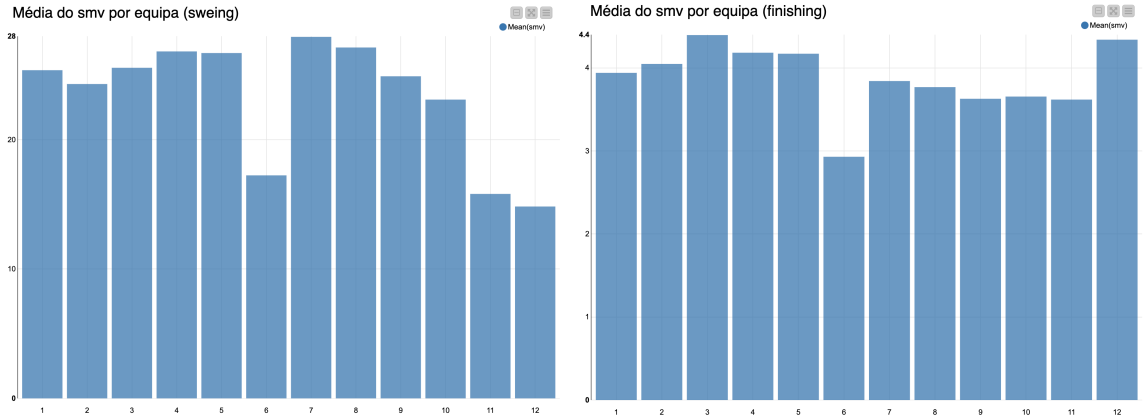


Figura 6: Média do SMV por equipa

Assim sendo, é de esperar que estes valores de média sejam influenciados por outros fatores como estes.

Através do desenvolvimento destes gráficos, aferimos também que a fábrica não está aberta às sextas-feiras, ou pelo menos que nenhum trabalhador comparece na fábrica:

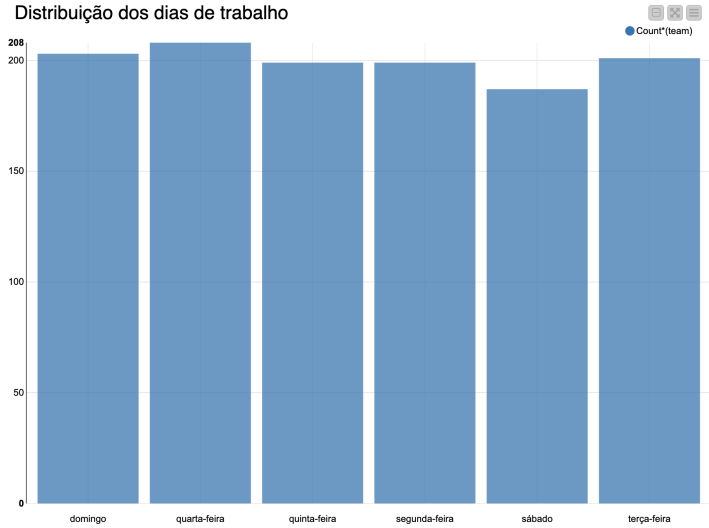


Figura 7: Distribuição dos dias da semana

Para acabar esta segunda fase da exploração de dados, decidimos investigar um bocado mais sobre a coluna *wip*, o único atributo no nosso conjunto de dados que contém *missing values*.

Usando a mesma estratégia que usamos para calcular a média do rendimento por equipa, descobrimos que todos os valores *NaN* da coluna *wip*, pertencem todos às entradas que correspondem ao departamento *finishing* (tivemos que fazer o tratamento dos valores em falta, que para fins de visualização, passamos todos ao valor fixo 0):

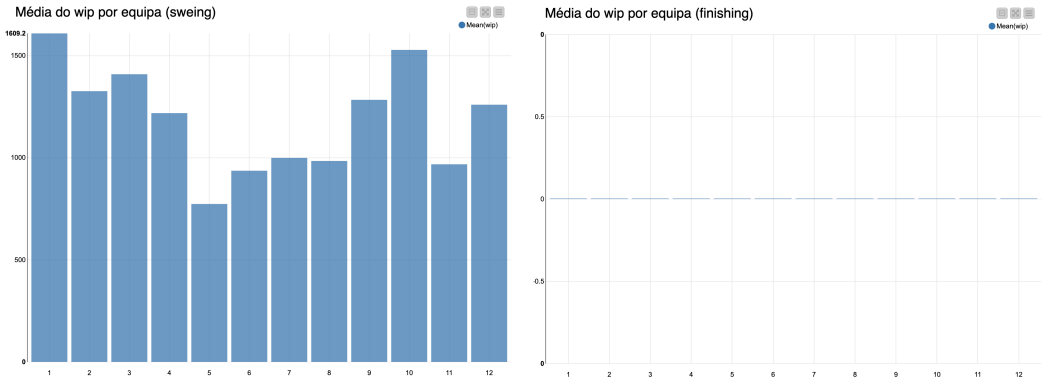


Figura 8: Média do WIP por equipa

Foi através destes gráficos que chegamos a uma das conclusões mais importantes em relação a este conjunto de dados. Os valores *NaN* da coluna *wip*, têm efetivamente um significado real, isto é, o departamento *finishing* fica à espera que o departamento *sweing* lhes envie os produtos que necessitam de acabamento, sendo este o verdadeiro dado que esta coluna representa: A quantidade de itens por linha que necessitam de ir para a segunda fase da linha de produção da fábrica.

Tentamos ainda definir relações entre atributos, usando *scatter plots*, *line plots* entre outras coisas, mas os resultados foram sempre desapontantes. Conseguimos, no entanto, definir uma relação entre os atributos *wip* e *no_of_style_change* e também entre o *wip* e o *no_of_workers*. Resumindo, quantas mais mudanças houveram no estilo do item e quantos mais trabalhadores houver na equipa, maior o valor do *wip*, isto pode significar que as linhas que necessitam de fabricar mais peças, para irem para segunda fase, têm mais trabalhadores e que quantas mais mudanças no estilo do produto houver, mais itens vão necessitar de acabamento.

3.2 Income Evaluation

Para este *dataset* começamos por mostrar várias distribuições de dados, pois ao contrário do conjunto de dados anterior, não faz sentido olhar para a média da nossa variável dependente, visto ser uma variável categórica.

Seguem-se algumas das distribuições mais pertinentes que desenvolvemos:

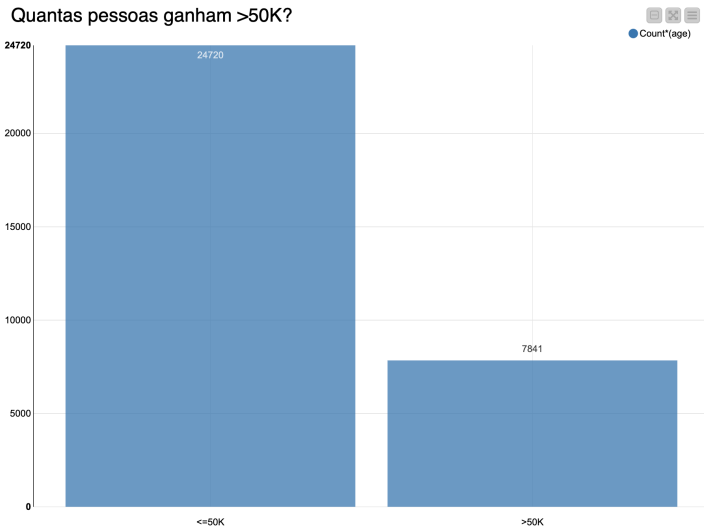


Figura 9: Distribuição do income

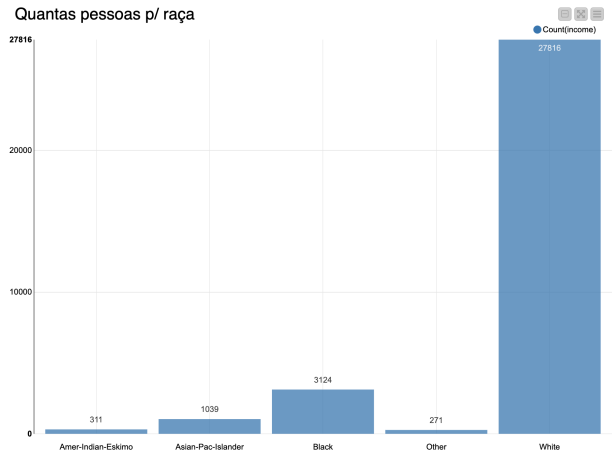


Figura 10: Distribuição da raça

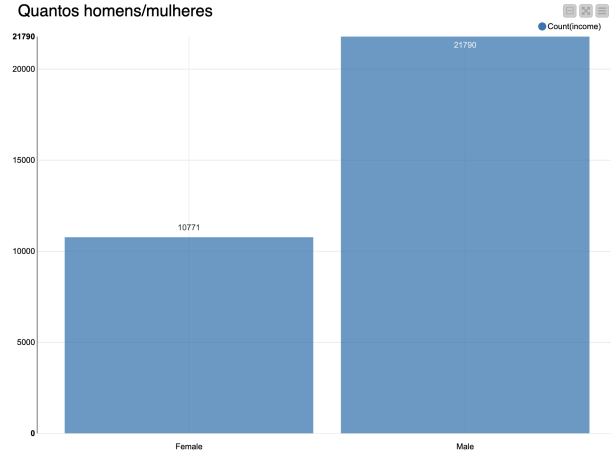


Figura 11: Distribuição do género

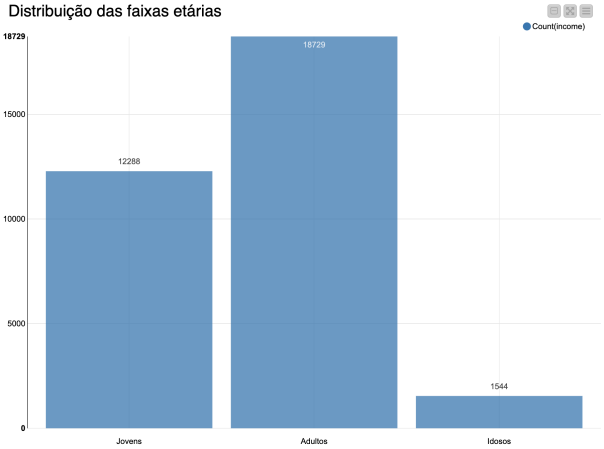


Figura 12: Distribuição da idade

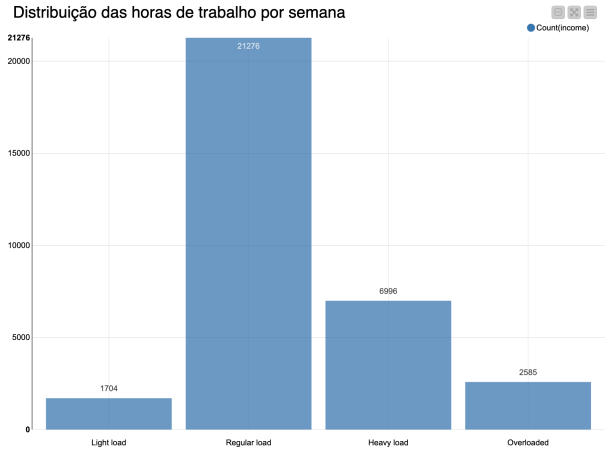


Figura 13: Distribuição das horas de trabalho

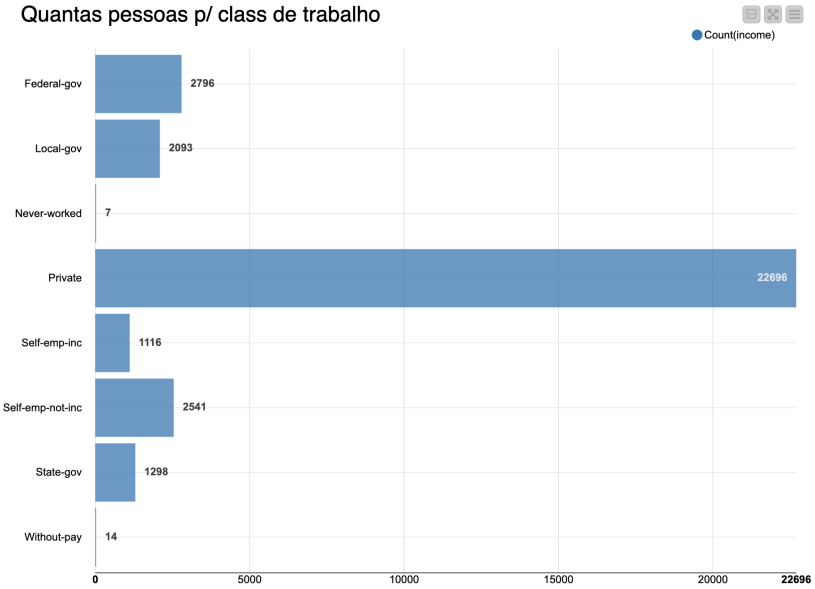


Figura 14: Distribuição da classe de trabalhador

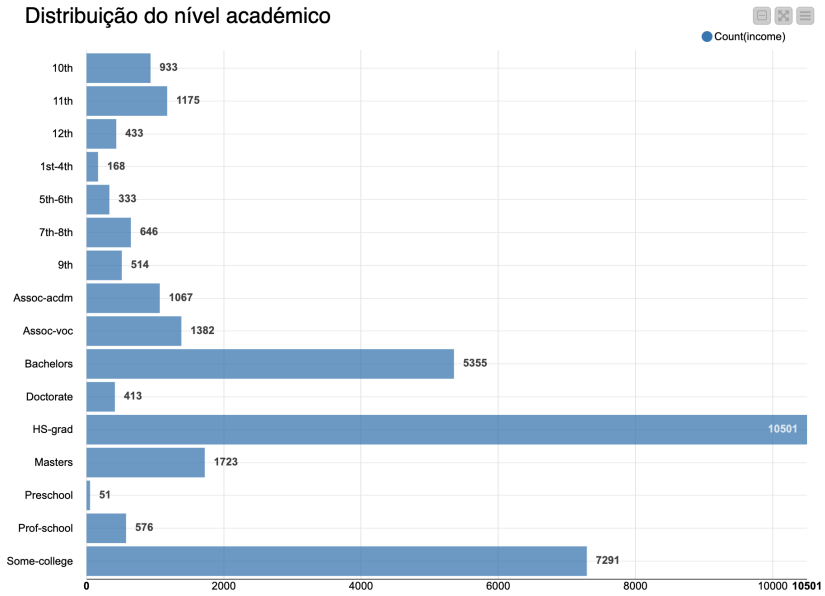


Figura 15: Distribuição do nível de educação

Mostramos ainda qual o valor de capital médio para cada classe de *income*, a distribuição geográfica e do estado civil.

Nesta fase, apenas por mera curiosidade, desenvolvemos alguns destes gráficos para mostrar como é que se distribuía a população portuguesa neste conjunto de dados. Os resultados foram os seguintes:

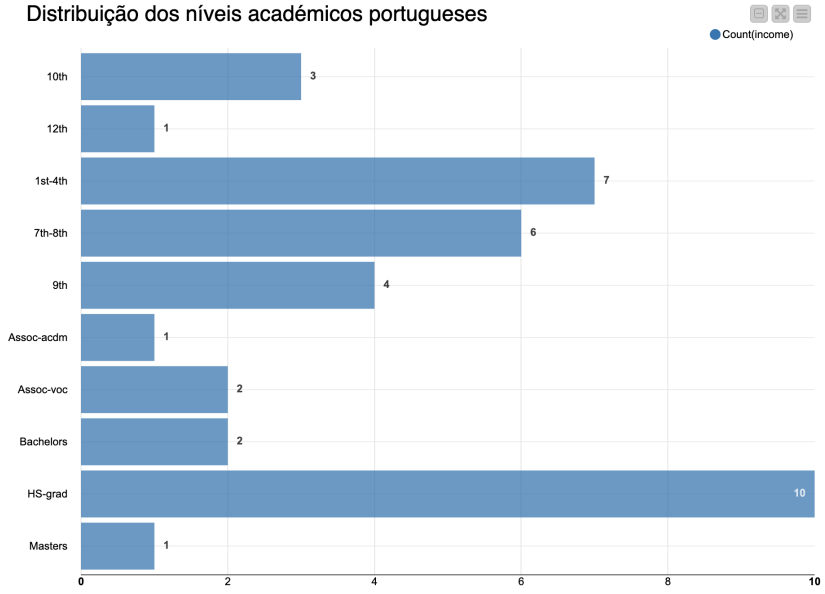


Figura 16: Distribuição do nível de educação dos portugueses

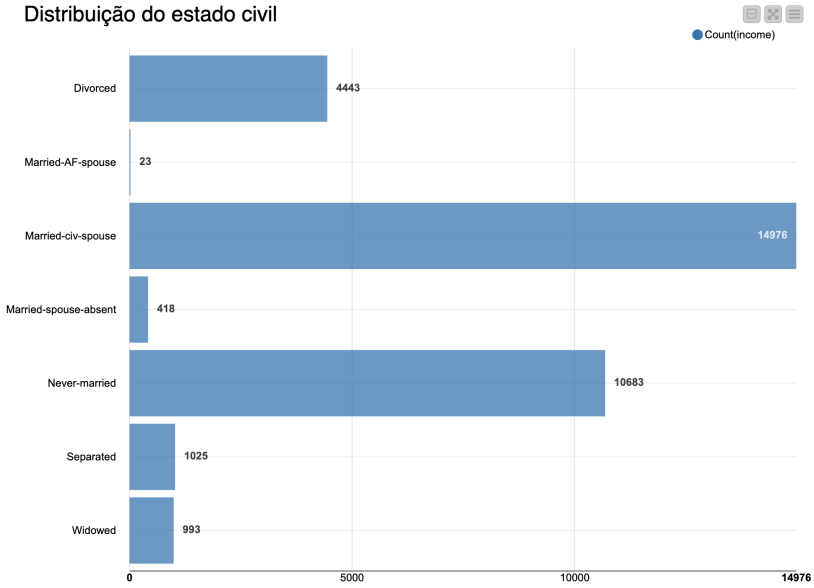


Figura 17: Distribuição do estado civil dos portugueses

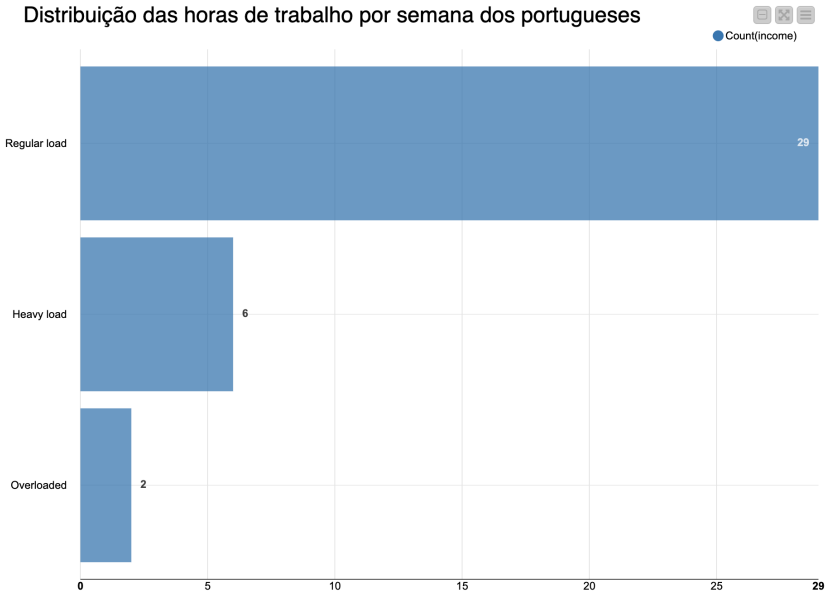


Figura 18: Distribuição do número de horas de trabalho semanal dos portugueses

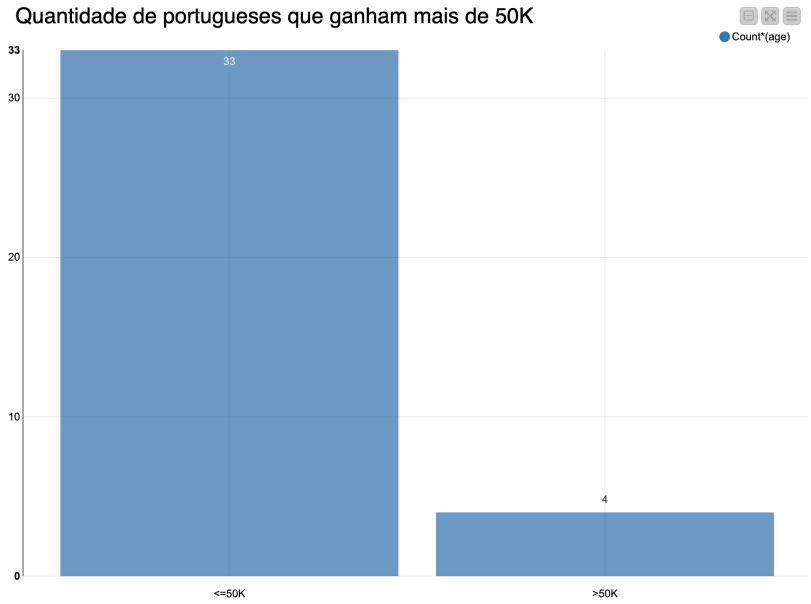


Figura 19: Distribuição do income dos portugueses

Na verdade, os portugueses parecem representar a população do conjunto de dados bastante bem. Embora não se consiga retirar nenhuma informação relevante diretamente destes gráficos, foi com base nestes que baseamos as nossas seguintes decisões.

Para este *dataset*, decidimos então desenvolver vários gráficos relacionados com a média do income pelos diferentes atributos, na tentativa de encontrar um padrão que nos revelasse quais as características mais comum à população que ganha mais de 50000$.

Entre os vários gráficos que desenvolvemos, dois deles saem destacados:

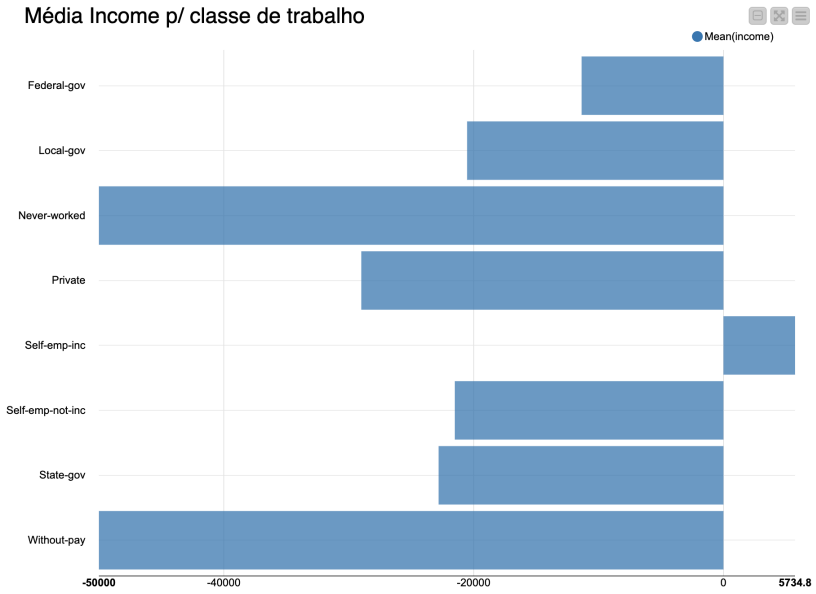


Figura 20: Média do income português por classe de trabalhador

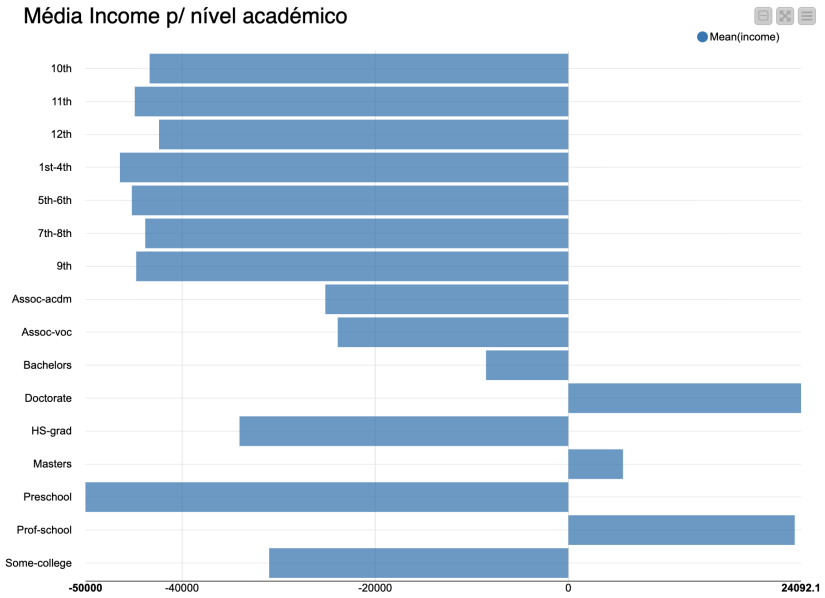


Figura 21: Média do income português por nível educacional

Repare que, para efeitos de visualização, tivemos que converter os valores nominais de *income* ">50K" e "<=50K" para 50000 e -50000, respetivamente, ou seja, -50000 representa alguém que recebe menos de 50000$.

No entanto, como nós agregamos os dados pela média do *income*, agora podemos ver a moda de cada entrada (olhando para o sinal do resultado) e aferir qual o número de pessoas que ganham mais de 50000$, pois quanto mais próximo de 0 for o resultado, menor será este número.

4 Modelos de aprendizagem

Esta é a fase final do trabalho, onde desenvolvemos modelos de aprendizagem supervisionada e não supervisionada, com o objetivo de fazer com que a máquina preveja o resultado das nossas variáveis dependentes.

Para cada um dos modelos que desenvolvemos, fizemos uma sub-divisão entre tratamento de dados, modelo e análise dos resultados. Desta forma, podemos organizar melhor o nosso *workflow*, transformando o tratamento de dados e o modelo em metanodos.

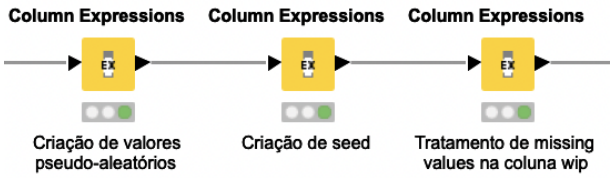
No metanodo do modelo, focamo-nos na forma como fazemos a partição dos dados. Em todos os modelos, fazemos a verificação entre *Hold-out Validation* e *Cross-Validation* e testamos para qual destes dois métodos a precisão do modelo é superior.

É também importante dizer que, para efeitos de comparar os resultados mais tarde, começamos por fazer o tratamento de dados mínimo possível para podermos avaliar e registar o valor de precisão inicial e mais tarde compará-lo à nossa tentativa de otimização. Assim, definimos então uma hierarquia, onde a linha mais abaixo corresponde sempre ao tratamento inicial.

O último pormenor a ser discutido é que os nodos que interrompem o fluxo e se encontram mais a cima, são aqueles que nós tentamos tratar os dados com, mas só pioraram o valor da precisão.

4.1 Produção e Vestuário

Para este conjunto de dados, para além do tratamento de dados inicial, definimos sempre dois fluxos, que diferem apenas na maneira como tratámos dos *missing values* do atributo *wip*. Esta limpeza foi feita então de 2 formas diferentes: a um valor fixo zero, ou então com valores pseudo-aleatórios. Para tal, usámos estes 3 nodos:



No primeiro nodo criamos uma nova coluna a partir das colunas *incentive* e *smv*. Se a coluna *incentive* fosse 0, o valor da nova coluna passaria a ser o de *smv* * 6. Se não iria manter o valor. No nodo seguinte, queriamos uma *seed*, que é o novo valor calculado anteriormente multiplicado pelo número da respetiva equipa, multiplicado por 10 novamente e calculado o resto da divisão inteira por 1501 (doutra forma seria possível dar 0 outra vez).

Falemos agora de cada um dos modelos desenvolvidos pelo grupo:

4.1.1 Modelo de Regressão Linear

A regressão linear é uma técnica de aprendizagem supervisionada usada para prever variáveis do tipo contínuo.

Para o tratamento de dados inicial, começamos por passar a nossa variável dependente, *actual_productivity*, para double, retiramos a coluna *rowID*, que não acarreta nenhuma importância para o nosso modelo e fizemos a limpeza de valores da coluna *wip* com o fix value zero. Para este tratamento, registamos $R^2 = 0.143$.

Para otimizar este valor, começamos por corrigir alguns valores que não faziam sentido, como por exemplo valores do tipo Double nas colunas *wip* e *no_of_workers*, que deviam ser representados como inteiros. Fizemos ainda o parse da nossa coluna *date*, para identificar o dia e o mês de forma isolada. Nesta fase reparamos que a coluna *targeted_productivity* tinha um valor 0.07, que nós assumimos ser uma gralha, então fizemos a conversão desse valor para 0.7, um valor já existente na lista de valores únicos do atributo. Para terminar, tratamos dos *outliers* das colunas *incentive* e *actual_productivity*, pois são as únicas que sobem a precisão do modelo, substituindo-os pelo próximo valor permitido. No entanto, reparámos que para tratar dos *outliers* do limite superior da nossa variável dependente, iamos precisar de outro nodo, pois a ferramente não estava a captar valores a cima de 1 como *outliers*. No entanto, como estamos a considerar este valor como uma percentagem, não faria sentido termos rendimentos superiores a 100%. Para terminar, reparamos que a nossa coluna *department* tinha alguns valores errados, umas gralhas, então corrigimo-los para o nome correto. Ainda tentamos normalizar os atributos, usando *minimax*, *Z-Score* e normalização por escala decimal, mas todos estes métodos só parecem descer a precisão do nosso modelo.

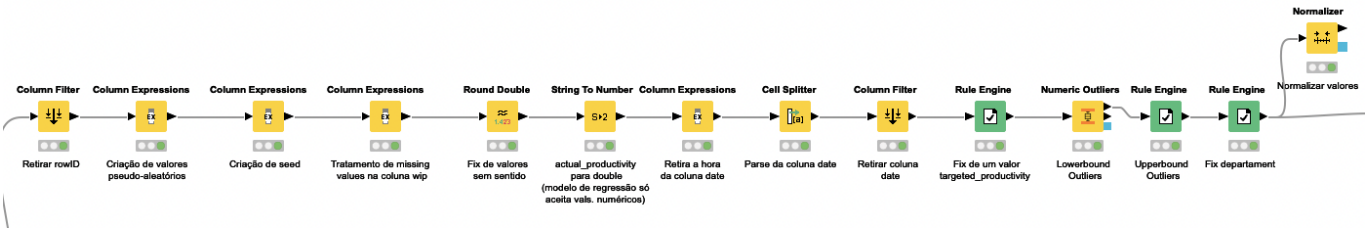


Figura 22: Tratamento de dados da regressão linear

4.1.2 Modelo de Classificação

A classificação é também uma ténica de aprendizagem supervisionada que é utilizada para prever variáveis de tipo categórica.

O tratamento de dados inicial que desenvolvemos para este modelo, é quase idêntico ao tratamento inicial do modelo de regressão linear. A principal diferença é que aqui temos que criar grupos nominais para a nossa variável dependente. Para este tratamento, obtivemos uma precisão de 50%.

```
$actual_productivity$ < 0.4 => "bad"
$actual_productivity$ >= 0.4 AND $actual_productivity$ < 0.5 => "unsatisfatory"
$actual_productivity$ >= 0.5 AND $actual_productivity$ < 0.7 => "satisfatory"
$actual_productivity$ >= 0.7 AND $actual_productivity$ < 0.8 => "good"
$actual_productivity$ >= 0.8 => "very good"
```

Figura 23: Grupos nominais para o modelo de classificação

Otimizando agora o nosso modelo da mesma forma que fizemos para o modelo de regressão linear, reparámos que fazer o parse da coluna *date*, não ajudava em nada o modelo, apenas o piorava em alguns casos. Tentamos também normalizar atributos e tratar dos seus *outliers* mas ambas estas ténicas também não pareceram aumentar a precisão do modelo.

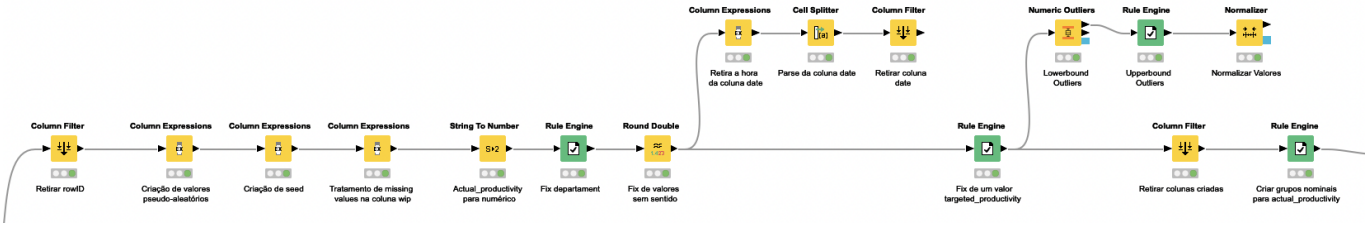


Figura 24: Tratamento de dados da regressão linear

4.1.3 Modelo de Regressão Logística

A regressão logística é a terceira e última técnica de aprendizagem supervisionada que implementámos neste *dataset*. Ela é usada para prever variáveis binárias, ou seja, variáveis que só possam tomar 2 valores. É, de certa forma, um caso particular da classificação.

Fizemos exatamente o mesmo tratamento de dados que o modelo de classificação, tanto inicialmente como aplicámos também a mesma otimização. Obtivemos assim uma precisão inicial de 27%. No entanto, tivemos que definir grupos nominais diferentes, pois a regressão logística só funciona para variáveis binárias:

```
$actual_productivity$ < $targeted_productivity$ => "unsatisfatory"
$actual_productivity$ >= $targeted_productivity$ => "satisfatory"
```

Figura 25: Grupos nominais para o modelo de regressão logística

A principal diferença entre a otimização deste modelo e do modelo de classificação, é que há melhorias nítidas por normalizar as colunas *smv* e *wip*, que são colunas que contém uma grande disparidade entre os seus extremos e a sua média.

Quanto aos restantes nodos, o modelo reagiu da mesma forma que a classificação, que era exatamente isso de que estávamos à espera.

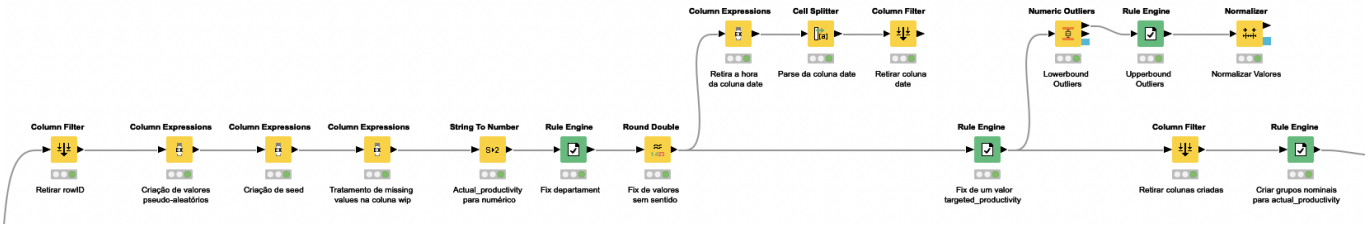


Figura 26: Tratamento de dados da regressão logística

4.1.4 Modelo de Segmentação

A segmentação é a primeira e única técnica de aprendizagem não supervisionada que vamos aplicar no nosso projeto. Estes modelos são usados para dividir um conjunto de dados em grupos (*clusters*) com base nas suas características comuns.

Para poder comparar valores e resultados, decidimos aplicar 2 formas de *clustering* diferentes. Recorremos aos algoritmos *k-means* e ao *k-medoids*, que são algoritmos cujo objetivo é particionar o conjunto de dados num número pré-definido de *clusters*.

O k-means é um algoritmo que tenta encontrar k centróides (ou médias) que minimizem a soma das distâncias euclidianas entre cada ponto de dado e o centróide do cluster ao qual ele é atribuído. Já o k-medoids é uma variação do k-means que utiliza medóides em vez de centróides. O medóide é o ponto de dados mais representativo de um cluster, ou seja, aquele que minimiza a soma das distâncias entre ele e os demais pontos do mesmo cluster.

Para além de usarmos 2 algoritmos para comparar resultados, definimos ainda mais 2 conjunto de modelos de segmentação utilizando cada vez mais *clusters*, para percebermos se neste caso era benéfico criar mais ou menos conjuntos de dados. Começámos então por definir k = 2, de seguida fizemos para k = 5 e por último k = 10. É também importante notar que aplicámos o tratamento de dados da regressão logística ao conjunto de dados com 2 *clusters* e o tratamento de dados da classificação aos restantes. Para estes tratamentos, obtivemos valores iniciais de 44%, de 18% e de 9%, respetivamente (esta percentagem refere-se à precisão do algoritmo que obteve melhores resultados para o mesmo número de *clusters*).

Estávamos à espera que estes modelos refletissem todos o mesmo comportamento que o modelo de regressão logística e o de classificação, respetivamente, mas não foi esse o caso. Enquanto os modelos com k = 5 e k = 10 reagiram de forma semelhante, mesmo que com resultados bastante piores, o modelo com k = 2 beneficiou até do tratamento de *outliers* e da normalização de alguns atributos. No entanto, o modelo com 10 *clusters* benefeciou do tratamento dos *outliers*, enquanto que no modelo com apenas 5 *clusters*, o tratamento de outliers não aumentou a precisão.

Os grupos nominais criados para k = 2 e k = 5 foram iguais aos modelos de regressão logística e de classificação. Seguem-se então as categorias criadas para o modelo com k = 10:

```
$actual_productivity$ < 0.2 => "Awfull"
$actual_productivity$ >= 0.2 AND $actual_productivity$ < 0.3 => "Very bad"
$actual_productivity$ >= 0.3 AND $actual_productivity$ < 0.4 => "Bad"
$actual_productivity$ >= 0.4 AND $actual_productivity$ < 0.5 => "Unsatisfatory"
$actual_productivity$ >= 0.5 AND $actual_productivity$ < 0.6 => "Decent"
$actual_productivity$ >= 0.6 AND $actual_productivity$ < 0.7 => "Satisfatory"
$actual_productivity$ >= 0.7 AND $actual_productivity$ < 0.8 => "Good"
$actual_productivity$ >= 0.8 AND $actual_productivity$ < 0.9 => "Very good"
$actual_productivity$ >= 0.9 AND $actual_productivity$ < 0.98 => "Amazing"
$actual_productivity$ >= 0.98 => "Perfection"
```

Figura 27: Grupos nominais para o modelo de regressão logística

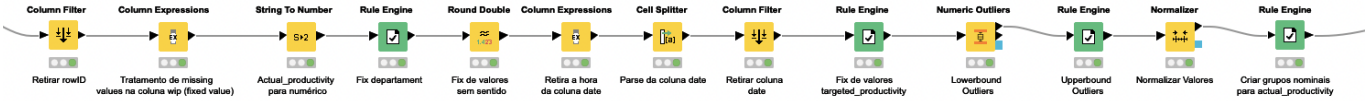


Figura 28: Tratamento de dados da segmentação com k = 2

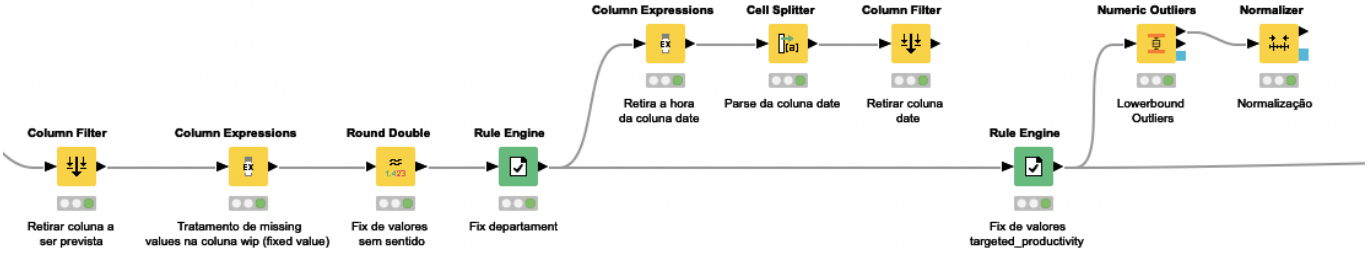


Figura 29: Tratamento de dados da segmentação com k = 5

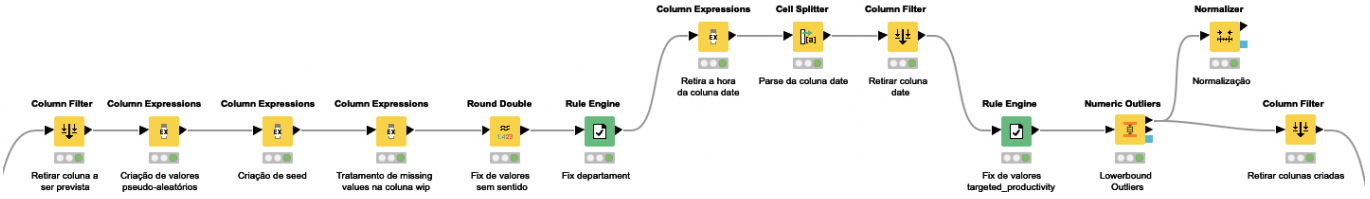


Figura 30: Tratamento de dados da segmentação com k = 10

4.2 Income Evaluation

No segundo conjunto de dados, a limpeza de *missing values* foi menos desafiante que a anterior.

Começamos por transformar todas as entradas que correspondiam a um "?" em valores em falta, da seguinte forma:

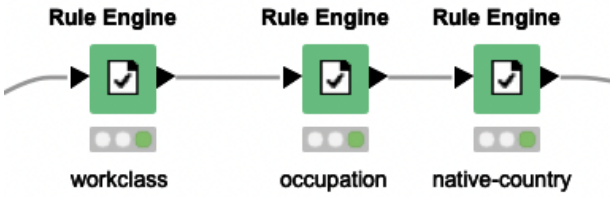


Figura 31: Transformação em *missing values*

```
NOT ($workclass$ = "?") => $workclass$
```

Figura 32: Exemplo de como transformar em *missing value*

Depois de transformados em valores em falta válidos para a ferramenta, aplicamos o nodo *Missing Values* para poder tratar de cada um dos atributos separadamente:

Figura 33: Tratamento dos valores em falta para cada atributo

Aplicámos este tratamento a todos os seguintes modelos que desenvolvemos, bem como lhes retirámos a coluna *fnlwgt*, pois esta coluna, como já foi explicado anteriormente, representa o peso da amostragem de cada entrada, não acrescentado qualquer tipo de informação relevante aos nossos modelos.

4.2.1 Modelo de Classificação

Começamos por desenvolver uma modelo de classificação visto que para este conjunto de dados a nossa variável dependente é categórica.

Inicialmente, conseguimos correr o modelo fazendo apenas a limpeza dos valores em falta, obtendo imediatamente uma ótima precisão de 80%.

O tratamento de dados deste modelo, apenas benefeciou do tratamento de *outliers* de um dos atributos, *hours-per-week*. Neste caso, visto que a nossa variável dependente já é do tipo nominal, nem se quer foi necessário criar as categorias à nossa medida.

Ainda tentamos normalizar valores, de várias formas diferentes, tentamos retirar algumas colunas que podiam aumentar a precisão do modelo por não fornecer informação interessante, assim como a coluna *fnlwgt* e ainda exploramos *feature engineering*, ou engenharia de recursos, que se baseia em criar novas colunas/atributos para tentar aumentar a precisão do modelo. Visto não termos experiência com esta técnica e os resultados não foram muito melhores, decidimos retirar este processo para não correr o risco de enviesar demasiado os dados.

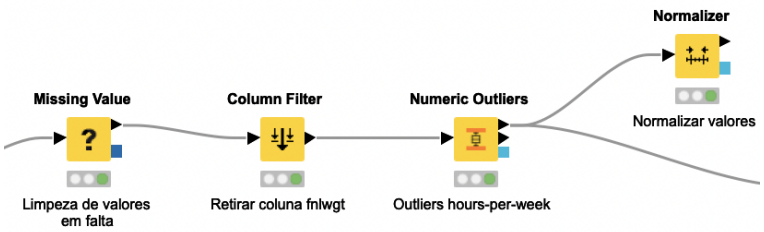


Figura 34: Tratamento dos valores em falta para cada atributo

4.2.2 Modelo de Regressão Logística

Desenvolvemos mais um modelo de regressão logística e, visto que a nossa variável dependente é uma variável binária, pois só pode tomar os valores ">50K" e "<=50K", decidimos aplicar exatamente o mesmo tratamento de dados que o modelo de classificação, na esperança que os resultados fossem exatamente os mesmos.

Para a nossa surpresa, o algoritmo obteve inicialmente apenas 71% de precisão, e apenas quando otimizamos o tratamento é que foi capaz de atingir o valor inicial do modelo de classificação. Este valor pode advir da forma como o sistema calcula as previsões ou simplesmente de azar nas partições dos conjuntos de dados.

4.2.3 Modelo de Segmentação

De forma análoga, desenvolvemos o mesmo modelo mas desta vez recorrendo a uma metodologia não supervisionada.

Mais uma vez, como a nossa variável é binária, decidimos aplicar o mesmo tratamento de dados ao correr os nossos algoritmos com apenas 2 *clusters*. Chegamos ao consenso de não desenvolver modelos com maior número de *clusters* como consequência dos resultados obtidos durante o desenvolvimento de modelos recorrendo à segmentação no conjunto de dados anterior.

Da mesma forma, voltamos a correr o mesmo tratamento para os algoritmos *k-means* e *k-medoids*, para ver se havia alguma diferença considerativa entre os dois.

Inicialmente, foi surpreendente o quão melhor o algoritmo *k-medoids* previu com precisão os valores, obtendo uns impressionante 77%. Depois de mexer em alguns hiperparâmetros e de alterar a partição dos dados, as precisões doos algoritmos começaram a aproximar-se.

Desta vez, tanto a normalização como o tratamento de *outliers*, independentemente de como são feitos, descem, e consideravelmente, a precisão do modelo.

4.2.4 Redes Neuronais Artificiais

Para acabar os modelos deste conjunto de dados, decidimos aplicar Redes Neuronais Artificiais, que são modelos computacionais inspirados no funcionamento do cérebro humano. Elas são compostas por várias camadas de unidades de processamento interconectadas, chamadas neurônios artificiais, que trabalham juntos para realizar tarefas de processamento de informações complexas, como reconhecimento de padrões, classificação, previsão e controle. Elas podem ser usadadas tanto para aprendizagem supervisionada como para aprendizagem não supervisionada, apenas difere na forma como o treino da rede é feito, com dados rotulados ou não rotulados. Ou seja, se em cada exemplo de entrada, a rede tem acesso ao *output* desejado, isso seria aprendizagem supervisionada e o inverso aplica-se à aprendizagem não supervisonada.

O primeiro passo para aplicar uma RNA, é escolher que tipo de arquitetura se adequa melhor ao nosso problema. Existem várias arquiteturas diferentes, cada uma com as suas vantagens e desvantagens para certas situações, mas nós optamos por usar MLP.

MLP - Multi Layer Perceptron

Multi Layer Perceptron é uma arquitetura de rede neural artificial do tipo *feedforward*. Essa arquitetura é composta por várias camadas de neurônios, cada uma delas com múltiplos neurônios que se comunicam com as camadas adjacentes, mas sem formar ciclos (daí o termo, que significa "avanço").

Infelizmente, a ferramenta em que estamos a desenvolver o projeto é um bocado limitadora no que toca a desenvolver Redes Neuronais, pois os únicos parâmetros que temos a liberdade de alterar são o número de camadas intermédias e o número de neurónios (fixo e igual) em cada camada.

É importante mencionar que as redes neuronais, embora possam ser capazes de receber todo o tipo de dados, no KNIME somos forçados a passar apenas valores do tipo Double que foram normalizados, isto é, entre 0 e 1. Assim sendo, retirámos todas as colunas que não sejam de tipo numérico, normalizamos as que são e obtivemos uma precisão inicial de 80%. Já seria de esperar que os resultados fossem muito bons, visto que a rede neuronal é muito boa a prever este tipo de resultados. Infelizmente isto também significa que otimizá-la é bastante mais complexo.

Dito isto, seria demasiado fácil retirar todas as colunas que não sejam do tipo numérico, portanto o grupo decidiu fazer uma espécie de conversão (extremamente básica) de String para numérico, que mais tarde aprendemos ser uma ténica chamada *embedding*. Mais tarde, fazendo todas as transformações necessárias, decidimos tratar de todos os *outliers* e, como não poderia deixar de ser, normalizar os valores de entrada da rede.

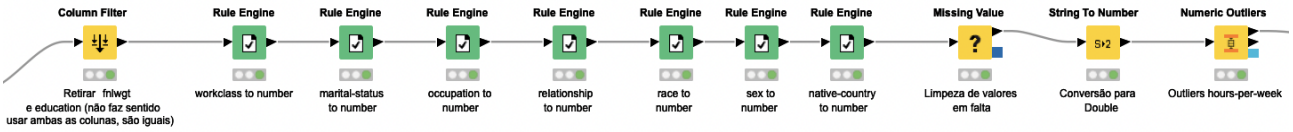


Figura 35: Tratamento de dados da Rede Neuronal - MLP

DL4J - DeepLearning4J Integration

O DeepLearning4J é uma extensão da ferramente que permite implementar redes neuronais com acesso a muitos mais hiperparâmetros, como o número de neurónios em cada camada, as funções de ativação, as funções de transferência e muito mais. Com esta vantagem de ter mais liberdade nos parâmetros, vem a desvantagem de ser muito mais complexa e da sua otimização ser ainda mais difícil.

Acabamos por fazer o mesmo tratamento de dados da RNA de arquitetura MLP, focamo-nos apenas nos hiperparâmetros que discutiremos mais para a frente.

5 Resultados Obtidos

Nesta secção final do relatório, vamos apresentar os resultados finais, de cada um dos modelos, após a otimização do tratamento de dados. É nesta parte que vamos também discutir que tipo de particionamento de dados resulta melhor para cada metodologia, entre a *Hold-out Validation* e a *Cross-Validation*, que foram os únicos tipos de partições que experimentámos.

A *Hold-out Validation* consiste em dividir o conjunto de dados em 2 partes: treino e teste. A *Cross-validation* é um bocado mais complexa, consiste em dividir o conjunto de dados em várias partes, experimentar cada uma dessas partes isoladamente e no final fazer a junção de todas essas partes para avaliar o resultado.

Independentemente do tipo de partição e modelo, usámos sempre uma random seed para ter a certeza que não obtemos nenhum bom resultado apenas por sorte e usamos também *stratified sampling*, que é útil para reduzir o erro da amostragem, dividindo o conjunto em grupos homogéneos e aplicando depois *random sampling*.

5.1 Produção e Vestuário

Neste conjunto de dados existe uma coisa comum a todos os modelos. Como dito anteriormente, nós fizemos sempre o tratamento de dados para dois tipos de limpeza de *missing values* diferentes, para um valor fixo ou para valores pseudo-aleatórios. Isto pois, como dito anteriormente, os *missing values* deste *dataset* apresentam um significado real.

O nosso objetivo com isto era testar para ver se havia diferença entre as duas formas de limpeza, o que não se verificou. Claro que alguns modelos benefeciaram dos valores pseudo-alteatórios, mas isso deve-se à forma como os cálculos são feitos e, mesmo assim, o benefício era mínimo. No entanto, deixamos sempre o fluxo que aumentava mais a precisão como o tratamento de dados final.

5.1.1 Modelo de Regressão Linear

Precisão Inicial do Modelo: 0.143

Através da otimização do tratamento de dados, conseguimos aumentar em quase 4x o valor da precisão. Estávamos à espera de mais na verdade, porém quando experimentamos engenharia de recursos, a precisão do modelo aumentou exponencialmente, indicando que poderia estar a ocorrer *overfitting*, que é quando um modelo se ajusta demasiado aos dados de treino, memorizando o ruído e as flutuações dos dados, tendo depois dificuldades a identificar padrões nos dados de teste.

O tipo de partição que mais benefícia este modelo é a *Hold-out Validation*.

Em seguida, os resultados do modelo de regressão:

R^2:	0,449
Mean absolute error:	0,095
Mean squared error:	0,017
Root mean squared error:	0,131
Mean signed difference:	-0,002
Mean absolute percentage error:	0,155
Adjusted R^2:	0,449

Figura 36: Resultados modelo de regressão linear

5.1.2 Modelo de Classificação

Precisão Inicial do Modelo: 50%

O modelo de classificação aumentou consideravelmente com a otimização, mas é importante notar que às vezes, dependendo da partição, tinha picos muito baixos (ou até muito altos) de precisão. Mas 70% é um bom valor para representar a média da precisão do nosso modelo.

Scorer View

Confusion Matrix

	bad (Predicted)	good (Predicted)	satisfatory (Pr…	unsatisfatory (…	very good (Pr…	
bad (Actual)	2	3	7	2	6	10.00%
good (Actual)	0	41	2	0	24	61.19%
satisfatory (Ac…	1	7	33	1	16	56.90%
unsatisfatory (…	1	2	6	0	4	0.00%
very good (Ac…	0	1	5	2	134	94.37%
	50.00%	75.93%	62.26%	0.00%	72.83%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
70.00%	30.00%	0.527	210	90

Figura 37: Resultados modelo de classificação

5.1.3 Modelo de Regressão Logística

Precisão Inicial do Modelo: 27%

Este modelo foi daqueles, ou até mesmo aquele que teve um maior aumento de precisão através da otimização do tratamento de dados. Foi também o primeiro modelo onde a *Cross-validation* é o tipo de partição mais benéfico.

Scorer View

Confusion Matrix

	satisfatory (Predicted)	unsatisfatory (Predicted)	
satisfatory (Actual)	818	56	93.59%
unsatisfatory (Actual)	255	68	21.05%
	76.23%	54.84%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
74.02%	25.98%	0.182	886	311

Figura 38: Resultados modelo de regressão logística

5.1.4 Modelo de Segmentação

Como dito anteriormente, desenvolvemos três modelos usando segmentação, com número de *clusters* diferentes. Em cada um dos três modelos, para o mesmo tratamento de dados, aplicamos os algoritmos *k-means* e *k-medoids* para poder comparar os valores entre os dois.

Para podermos avaliar os resultados dos modelos, usando o nodo Scorer, decidimos fazer o "join"do *dataset* original com o *dataset* tratado. Assim podemos comparar a nossa coluna *Cluster* à nossa variável dependente.

A figura que se segue ilustra como é que organizamos o nosso workflow. A linha de cima refere-se ao algoritmo *k-means* e a de baixo ao *k-medoids*. No meio, encontra-se um metanodo que corresponde ao tratamento de dados necessário para podermos comparar os valores ao conjunto de dados original, ou seja, à criação dos grupos nominais da nossa variável dependente contínua.

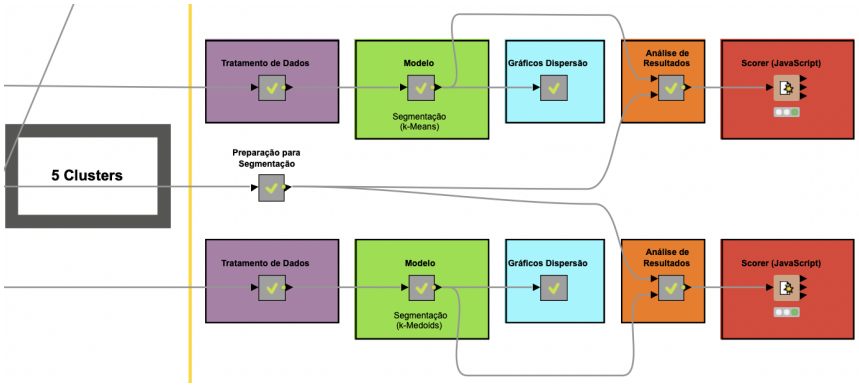


Figura 39: Workflow dos modelos de segmentação

Precisão Inicial do Modelo...

k-means

- k = 2 : 43%
- k = 5 : 18%
- k = 10 : 9%

k-medoids

- k = 2 : 44%
- k = 5 : 16%
- k = 10 : 8%

O nosso objetivo com o desenvolvimento de vários modelos com número de *clusters* diferentes, era avaliar o benefício de ter mais ou menos conjuntos. Assim, podemos dizer com certeza que quantos mais *clusters* mais difícil se torna para o modelo fazer a previsão correta ou, pelo menos, mais complexo se torna o tratamento de dados.

Reparámos também que a metodologia de aprendizagem não supervisionada através da segmentação, não é nada adequada para a previsão da variável dependente deste modelo, pois seria de esperar melhores valores para k = 2.

Sobre os algoritmos, a diferença entre o *k-means* e o *k-medoids* é quase obsoleta. Aquilo que não é obsoleto é a velocidade de cada algoritmo. Enquanto o *k-means* é extremamente rápido, o algoritmo *k-medoids* demora imenso tempo a correr, especialmente se usarmos *cross-validation* em vez de *hold-out validation*. Também se torna muito difícil fazer testes sobre o algoritmo *k-medoids*, pois sempre que fazemos uma alteração no modelo o nome dos *clusters* mudam, ao contrário do *k-means*.

As figuras que se seguem mostram os resultados otimizados de cada um dos modelos:

Scorer View

Confusion Matrix

	satisfatory (Predicted)	unsatisfatory (Predicted)	
satisfatory (Actual)	432	408	51.43%
unsatisfatory (Actual)	178	179	50.14%
	70.82%	30.49%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
51.04%	48.96%	0.013	611	586

Figura 40: Resultados modelo de segmentação 2-means

Scorer View

Confusion Matrix

	satisfatory (Predicted)	unsatisfatory (Predicted)	
satisfatory (Actual)	408	432	48.57%
unsatisfatory (Actual)	226	131	36.69%
	64.35%	23.27%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
45.03%	54.97%	-0.126	539	658

Figura 41: Resultados modelo de segmentação 2-medoids

Scorer View

Confusion Matrix

	bad (Predicted)	good (Predicted)	satisfatory (Pr…	unsatisfatory (…	very good (Pr…	
bad (Actual)	4	5	5	0	8	18.18%
good (Actual)	7	18	35	0	31	19.78%
satisfatory (Ac…	11	6	21	0	27	32.31%
unsatisfatory (…	2	0	9	0	4	0.00%
very good (Ac…	27	34	31	1	74	44.31%
	7.84%	28.57%	20.79%	0.00%	51.39%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
32.50%	67.50%	0.050	117	243

Figura 42: Resultados modelo de segmentação 5-means

Scorer View

Confusion Matrix

	bad (Predicted)	good (Predicted)	satisfatory (Pr…	unsatisfatory (…	very good (Pr…	
bad (Actual)	26	18	5	24	8	32.10%
good (Actual)	105	56	34	40	31	21.05%
satisfatory (Ac…	58	32	30	57	55	12.93%
unsatisfatory (…	20	5	7	16	4	30.77%
very good (Ac…	142	81	89	115	139	24.56%
	7.41%	29.17%	18.18%	6.35%	58.65%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
22.31%	77.69%	0.047	267	930

Figura 43: Resultados modelo de segmentação 5-medoids

Scorer View

Confusion Matrix

	Amazi…	Awfull…	Bad (P…	Decen…	Good …	Perfec…	Satisf…	Unsati…	Very b…	Very g…	
Amazi…	17	10	16	7	11	12	32	22	11	9	11.56%
Awfull…	0	0	0	0	0	0	0	0	0	0	undefined
Bad (…	5	13	6	2	11	3	15	11	4	11	7.41%
Decen…	11	12	5	8	11	4	11	10	6	5	9.64%
Good …	40	41	25	8	43	22	30	29	14	14	16.17%
Perfec…	2	8	2	2	12	9	4	11	3	3	16.07%
Satisf…	18	14	7	12	21	13	22	13	14	15	14.77%
Unsati…	8	1	7	3	10	4	6	9	1	3	17.31%
Very b…	0	0	0	0	0	0	0	0	0	0	undefined
Very g…	23	45	43	18	41	32	44	47	37	33	9.09%
	13.71%	0.00%	5.41%	13.33%	26.88%	9.09%	13.41%	5.92%	0.00%	35.48%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
12.28%	87.72%	0.023	147	1050

Figura 44: Resultados modelo de segmentação 10-means

Scorer View

Confusion Matrix

	Amazi…	Awfull…	Bad (P…	Decen…	Good …	Perfec…	Satisf…	Unsati…	Very b…	Very g…	
Amazi…	33	6	0	0	0	36	31	40	1	0	22.45%
Awfull…	0	0	0	0	0	0	0	0	0	0	undefined
Bad (…	3	0	10	28	2	30	1	7	0	0	12.35%
Decen…	7	0	22	18	0	19	0	14	3	0	21.69%
Good …	14	14	50	17	0	33	3	19	86	30	0.00%
Perfec…	12	0	0	0	0	7	28	9	0	0	12.50%
Satisf…	8	0	46	14	1	58	1	11	7	3	0.67%
Unsati…	1	0	11	17	0	18	1	4	0	0	7.69%
Very b…	0	0	0	0	0	0	0	0	0	0	undefined
Very g…	15	94	13	4	0	56	4	50	59	68	18.73%
	35.48%	0.00%	6.58%	18.37%	0.00%	2.72%	1.45%	2.60%	0.00%	67.33%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
11.78%	88.22%	0.049	141	1056

Figura 45: Resultados modelo de segmentação 10-medoids

5.2 Income Evaluation

Neste *dataset*, o tratamento de dados foi essencialmente igual para todos os modelos. Era um conjunto de dados mais básico que não requeria muitas alterações, tanto que grande parte das que fizemos apenas baixou a precisão. Portanto, entraremos mais a fundo nas redes neuronais artificiais, expondo simplesmente os resultados dos modelos desenvolvidos otimizados.

5.2.1 Modelo de Classificação

Precisão Inicial do Modelo: 80%

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	23123	1597	93.54%
>50K (Actual)	2825	5016	63.97%
	89.11%	75.85%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
86.42%	13.58%	0.608	28139	4422

Figura 46: Resultados modelo de classificação

5.2.2 Modelo de Regressão Logística

Precisão Inicial do Modelo: 71%

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	6836	608	91.83%
>50K (Actual)	1300	1025	44.09%
	84.02%	62.77%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
80.47%	19.53%	0.400	7861	1908

Figura 47: Resultados modelo de regressão logística

5.2.3 Modelo de Segmentação

Precisão Inicial do Modelo...

- k-means : 60%
- k-medoids : 77%

Nos modelos de segmentação deste conjunto de dados, aquilo que mais influenciou a precisão do modelo, seja pela positiva ou negativa, foi a forma como repartimos os dados, usando o algoritmo *k-means*. Inicialmente, pensavamos que o *k-medoids* era bastante melhor para este caso, mas quando fizemos o *stratified sampling* a precisão do primeiro algoritmo aumentou em quase 20% imediatamente.

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	7461	0	100.00%
>50K (Actual)	2253	55	2.38%
	76.81%	100.00%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
76.94%	23.06%	0.036	7516	2253

Figura 48: Resultados modelo de segmentação k-means

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	24326	394	98.41%
>50K (Actual)	6254	1587	20.24%
	79.55%	80.11%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
79.58%	20.42%	0.250	25913	6648

Figura 49: Resultados modelo de segmentação k-medoids

5.2.4 Redes Neuronais Artificiais

MLP

Precisão Inicial do Modelo: 80%

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	23108	1612	93.48%
>50K (Actual)	4300	3541	45.16%
	84.31%	68.72%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
81.84%	18.16%	0.438	26649	5912

Figura 50: Resultados da Rede Neuronal Artificial - MLP

DL4J

Precisão Inicial do Modelo: 45%

Usando agora uma extensão de redes neuronais artificiais em vez do nodo MLP do KNIME, temos muita mais liberdade para alterar os hiperparâmetros da rede.

A primeira grande alteração que fizemos foi alterar a função de atualização dos pesos de ADAGRAD para SGD, que não só aumentou imenso a precisão do nosso modelo, como também resolveu-nos o *undefined* na previsão de valores "¿50K"que ocorria quando o modelo não fazia nenhuma previsão desse tipo.

Decidimos correr a rede com apenas 2 camadas intermédias e 5 neurónios em cada camada, mas percebemos que com esta nova função de atualização dos pesos, a precisão aumentava por cada camada extra que colocáva-mos.

Tentamos ainda alterar a regularização da rede e aplicar *gradient normalization* mas resultaram em erros que não sabemos como corrigir. Alterámos também os *learning rates* de cada neurónio e na rede, mas não se registou diferença nenhuma.

Depois de vários ajustes, ficamos bastantes satisfeitos com o valor final, principalmente a subida registada depois da rede ser otimizada:

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	7094	323	95.65%
>50K (Actual)	1592	760	32.31%
	81.67%	70.18%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
80.40%	19.60%	0.343	7854	1915

Figura 51: Resultados da Rede Neuronal Artificial - DL4J

6 Conclusão

Em conclusão, trabalhar com a plataforma KNIME foi uma experiência valiosa para explorar diferentes técnicas de modelagem de dados, incluindo aprendizagem supervisionada, não supervisionada e redes neuronais. Durante o projeto, pudemos aplicar essas técnicas em dois *datasets* diferentes e comparar seus resultados.

Com base nos resultados obtidos, é possível concluir que os modelos de aprendizagem supervisionada e não supervisionada foram capazes de aprender informações importantes nos *datasets* em questão. Isto pode indicar que os nossos modelos têm potencial para serem aplicados noutras áreas.

Além disso, é importante destacar a importância do tratamento de dados no processo de modelagem. A qualidade dos dados é um fator crucial para o sucesso dos modelos, e um tratamento adequado pode melhorar significativamente a qualidade dos resultados.

Em suma, aprendemos imenso sobre *Machine Learning*, percebemos quais os melhores modelos a aplicar em determinadas situações e como otimizar a precisão de um modelo.