Universidade do Minho
Departamento de Informática

Aprendizagem e Decisão Inteligentes
Grupo 23

Artur Leite- A97027

Diana Teixeira- A95076

Diana Teixeira- A97516

Afonso Magalhães-A95250

Conteúdo

1. Introdução

No desenvolvimento deste trabalho prático, analisamos dois datasets distintos. Numa primeira fase, optamos por um dataset que contém informações relativas a classificação de salários. Construímos então um modelo de Machine Learning com o intuito de prever se o salário de um indivíduo excede os 50K ou não, tratando-se de problema de Classificação.

Já na segunda fase, devido ao nº do nosso grupo, foi-nos atribuído um dataset com informações sobre produção de vestuário, optando por desenvolver modelos de Machine Learning para prever a produção, desenvolvendo ambos modelos de Classificação e Regressão.

2. Classificação Salarial

2.1. Características do *Dataset*

Este *dataset* apresenta 32561 linhas e 15 colunas, as quais representam:

1. ***Age:*** Idade do indivíduo;
2. ***Workclass:*** Classe de trabalho do indivíduo;
3. ***Fnlwgt:*** Número de Identificação do indivíduo;
4. ***Education:*** Nível de educação do indivíduo;
5. ***Education.num****:* Nível de educação do indivíduo em forma numérica;
6. ***Marital-status:*** Estado matrimonial do indivíduo;
7. ***Ocupation:*** Tipo geral de ocupação do indivíduo;
8. ***Relationship:*** Estado de relacionamento do indivíduo;
9. ***Race:*** Raça do indivíduo;
10. ***Sex:*** Sexo biológico do indivíduo;
11. ***Capital-gain:*** Capital ganho pelo indivíduo;
12. ***Capital-loss:*** Capital perdido pelo indivíduo;
13. ***Hours-per-week:*** Horas de trabalho semanais do indivíduo;
14. ***Native Country:*** País de origem do indivíduo;
15. ***Income:*** Classificação do salário do indivíduo (>50k ou <=50k)

Das quais temos:

- **Categóricas** -> *Workclass, Education, Marital-status, Occupation, Relationship, Race, Sex, Native-country, Income*

- **Contínuas** ->*Age, Fnlwgt, Education-num, Capital-gain, Capital-loss, Hoursper-week*

2.2. Análise do Dataset

2.2.1. Income

Ao analisarmos o seguinte pie chart, conseguimos aferir que 76% dos indivíduos estudados no dataset recebem menos ou igual a 50 k de salário, com os restantes 24% recebendo mais de 50k.

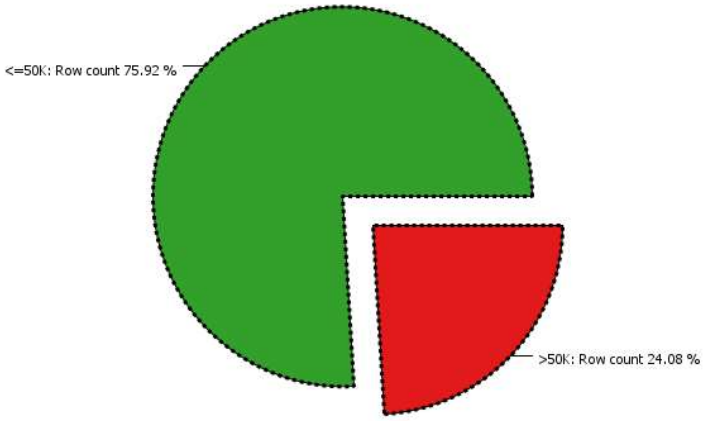


Figura 1- Distribuição da Classificação Salarial

2.2.2. Age

O intervalo de idades dos indivíduos presentes neste dataset varia entre os 17 e os 90 anos. Conseguimos aferir através do bar chart seguinte algumas informações, nomeadamente:

- Maior densidade de indivíduos entre os 20 e os 47 anos;
- Em todas as faixas etárias, maior parte dos indivíduos recebe menos de 50k;
- Percentagem de indivíduos que recebem mais de 50k mais predominante nos indivíduos entre os 40-49 anos;

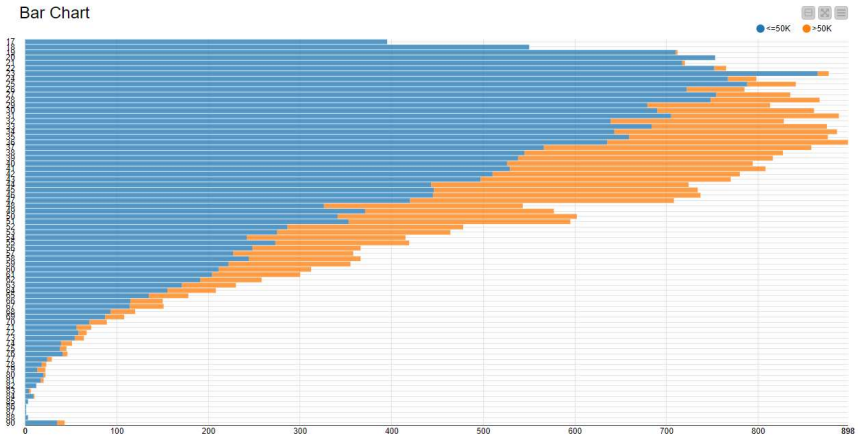


Figura 2- Bar chart das Idades

2.2.3. Workclass

Como podemos aferir no gráfico seguinte, grande maioria dos indivíduos trabalha no setor privado. Ainda assim, não existem grandes diferenças no que toca ao salário em maioria dos setores, com exceção da classe *self-emp-inc,* onde predominam os indivíduos que recebem maiores salários. Para além disso, deparamo-nos com a existência de alguns dados em falta, que terão de ser tratados futuramente.

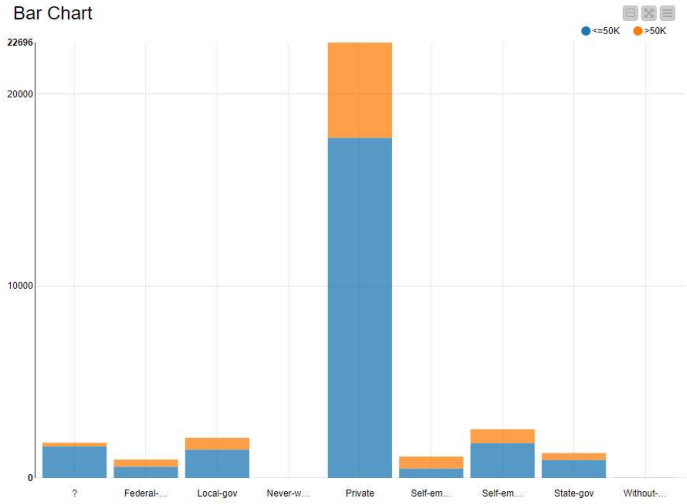


Figura 3-Bar chart da classe de trabalho

2.2.4. Education

Ao analisarmos os dados, chegamos à conclusão de que as colunas *Education* e *Education.num* são equivalentes. O mesmo número na coluna num correspondia sempre ao mesmo nível de educação e vice-versa. Analisando o seguinte gráfico, podemos aferir algumas informações:

- Grande parte dos indivíduos possui o nível de educação *HS-grad*;
- Geralmente, maior nível de educação resulta em maior número de indivíduos com melhor salário;
- Indivíduos com nível de educação *Masters* e *Doctorate* têm uma maior percentagem de salários maiores que 50k em relação a salários menores a 50k, ao contrário de todos os outros níveis de educação;

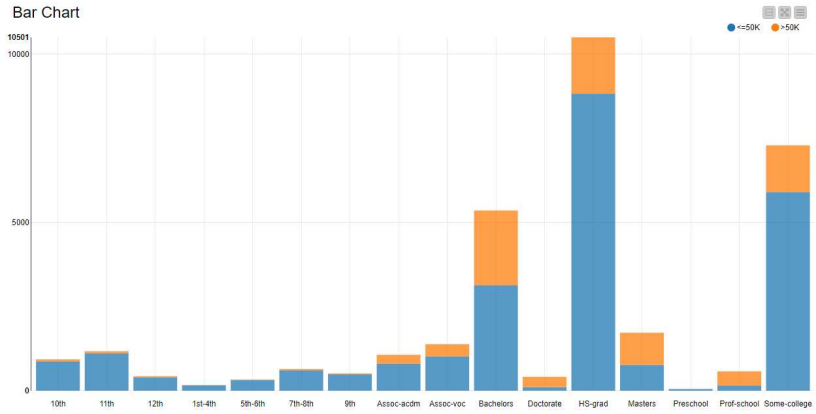


Figura 4-Bar chart de Educação

2.2.5. Marital Status

Ao analisarmos o gráfico seguinte, podemos ver que os indivíduos casados possuem melhores salários em comparação com os restantes.

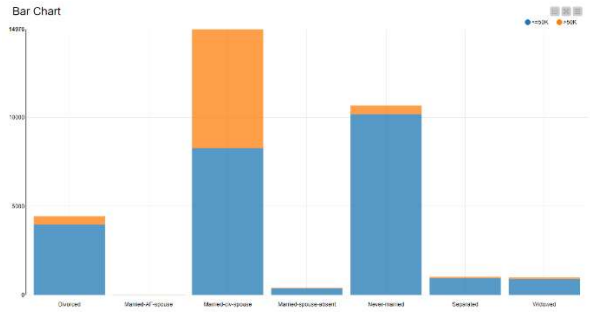


Figura 5-Bar chart do estado civil

2.2.6. Occupation

Analisando o gráfico seguinte, conseguimos aferir que a ocupação de um indivíduo afeta o seu salário significativamente. Para além disso, conseguimos ver que as ocupações de *Exec-managerial* e *Prof-specialty* oferecem os melhores salários, enquanto *Handlers-Cleaners* e *Farmers-Fishing* oferecem os piores salários.

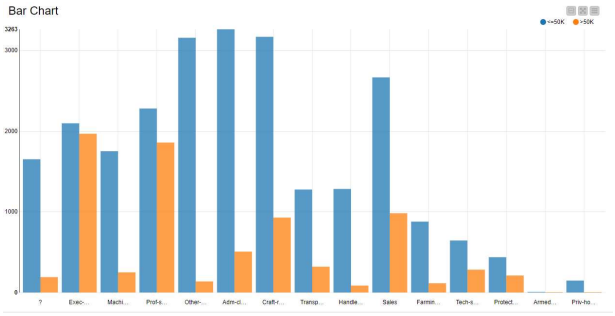


Figura 6- Bar chart da ocupação

2.2.7. Relationship

Analisando o seguinte gráfico, conseguimos ver que se confirma a conclusão retirada anteriormente de que os indivíduos casados recebem melhores salários.

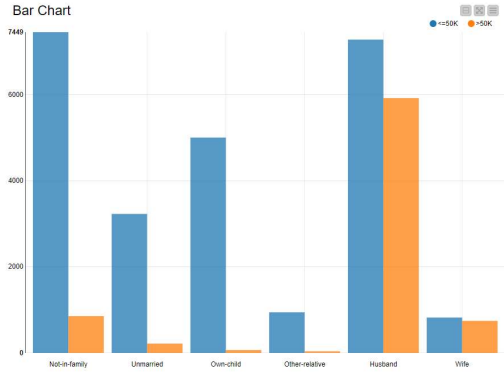


Figura 7- Bar chart do estado de relacionamento

2.2.8. Race

Analisando o seguinte gráfico, conseguimos aferir que foram estudados muitos mais indivíduos de raça branca neste dataset e que, geralmente, possuem melhores salários.

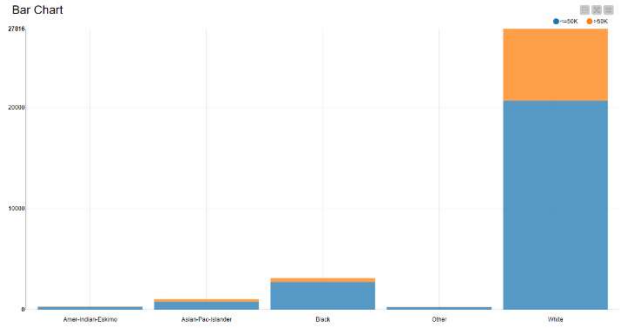


Figura 8-Bar chart da Raça

2.2.9. Sex

Analisando o seguinte gráfico, podemos aferir que maior parte dos indivíduos estudados neste dataset são do sexo masculino e possuem melhores salários que os indivíduos do sexo feminino.

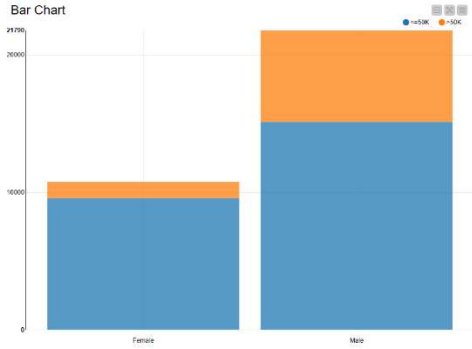


Figura 9- Bar chart de género

2.2.10. Hours per week

Ao analisar o gráfico seguinte, podemos ver que é difícil retirar algumas conclusões. Isto deve-se ao facto do número elevado de outliers estatísticos nesta vertente.

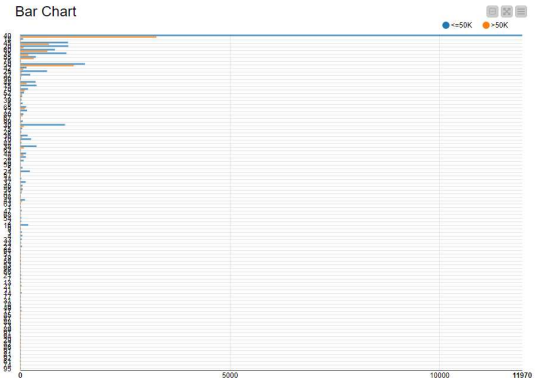


Figura 10-Bar chart das horas de Trabalho

2.3. Pré-Processamento dos Dados

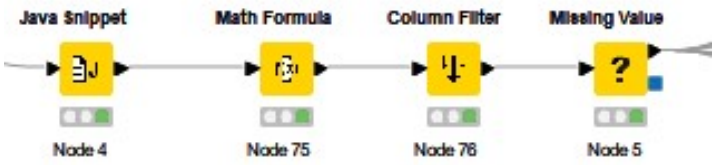


Figura 11- Pré-processamento de dados no Knime

Através do nodo **Java Snippet,** iniciamos o nosso processo de preparação de dados por substituir os elementos em falta, representados por um ponto de interrogação (?). O excerto de código presente no nodo é o seguinte:

```
out_nativecountry = c_country.equals("?") ? null : c_country;
out_workclass = c_class.equals("?") ? null : c_class;
out_occupation = c_occupation.equals("?") ? null : c_occupation;
```

Em seguida, notamos que o dataset original apresentava duas colunas relacionadas: *capital-gain* e *capital-loss*. Comparando as duas, reparamos que só existia *capital-gain* quando não existia *capital-loss* e vice-versa, sabendo isto, podemos juntar as duas colunas numa só coluna que iria ser populado pelo resultado da diferença de ambas (capital diff= capital gain – capital loss). Para este efeito utilizamos os nodos **Math Formula** seguido do **Column Filter**.

Finalmente, com o intuito de substituir todos os valores em falta, aplicamos o nodo **Missing Value,** onde substituímos valores do tipo **String** pelo valor mais frequente e elementos do tipo **Integer** pela média.

2.4. Modelação

2.4.1. Sem Tratamento adicional

Com o intuito de controlar a eficiência dos métodos que iremos utilizar, optamos por obter o resultado das previsões sem qualquer tipo de tratamento de dados adicional.

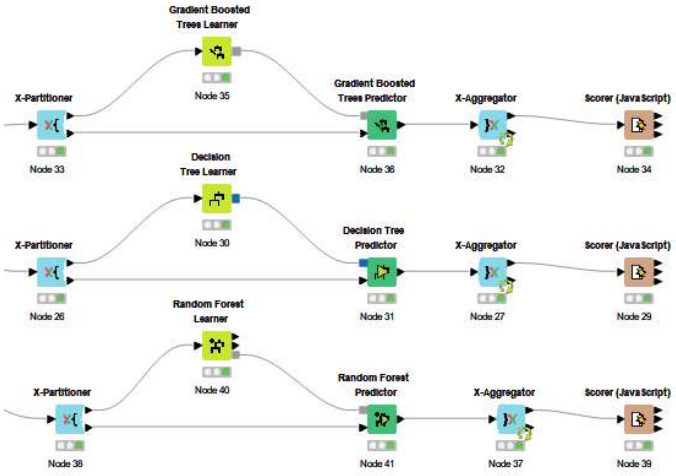


Figura 12- Testes sem tratamento de dados adicionais no Knime

Para serem criadas partições de dados para aprendizagem e teste, utilizamos os nodos de **X-Partitioner** e o **X-Aggregator.** Optamos por utilizar 10 validações e, dada a diferença significativa da percentagem de dados da coluna *Income,* utilizamos o método de *stratified sampling,* mantendo a proporção de elementos de cada classe salarial em cada partição. Devido ao facto de que este problema se trata de um problema de classificação, utilizamos os algoritmos de aprendizagem ***Gradient Boosted Trees, Decision Trees*** e ***Random Forest.*** Eis os resultados de cada um dos algoritmos:

- Gradient Boosted Trees

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	23347	1373	94.45%
>50K (Actual)	2781	5060	64.53%
	89.36%	78.66%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	23347	2781	5060	1373	94.45%	89.36%	94.45%	64.53%	91.83%
>50K	5060	1373	23347	2781	64.53%	78.66%	64.53%	94.45%	70.90%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
87.24%	12.76%	0.628	28407	4154

Figura 13- Scorer do Algoritmo de Gradient Boosted Trees

- Decision Trees

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	21990	2730	88.96%
>50K (Actual)	3066	4775	60.90%
	87.76%	63.62%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	21990	3066	4775	2730	88.96%	87.76%	88.96%	60.90%	88.36%
>50K	4775	2730	21990	3066	60.90%	63.62%	60.90%	88.96%	62.23%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
82.20%	17.80%	0.506	26765	5796

Figura 14- Scorer do algoritmo Decision Tree

- Random Forest

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	23401	1319	94.66%
>50K (Actual)	3024	4817	61.43%
	88.56%	78.50%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	23401	3024	4817	1319	94.66%	88.56%	94.66%	61.43%	91.51%
>50K	4817	1319	23401	3024	61.43%	78.50%	61.43%	94.66%	68.93%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
86.66%	13.34%	0.606	28218	4343

Figura 15- Scorer do Algoritmo Random Forest

Analisando os resultados, podemos ver que o algoritmo Gradient Boosted Trees é o mais eficiente, com uma *accuracy* de 87.24%. Os algoritmos de ***Decision Tree*** e ***Random Forest*** obteram uma *accuracy* de 82.20% e 86.66%. Numa primeira análise, estes resultados podem parecer bastante positivos, mas se analisarmos os valores de *Recall* e *Precision* para a class >50k, estes são significativamente baixos. Para resolvermos este prolema, iremos ter de adicionar alguns métodos de tratamento de dados.

2.4.2. Binning

Para tentar obter melhores alores de *Recall* e *Precision* optamos por utilizar o nodo de **Numeric Binner,** agrupando a coluna de idades por grupos mais pequenos, evitando termos imensas entradas diferentes. Agrupamos em 6 grupos:

- [0,20[anos;
- [20,30[anos;
- [30,40[anos;
- [40,50[anos;
- [50,60[anos;
- [60,+∞[anos.

O nosso modelo do Knime fica com este aspeto:

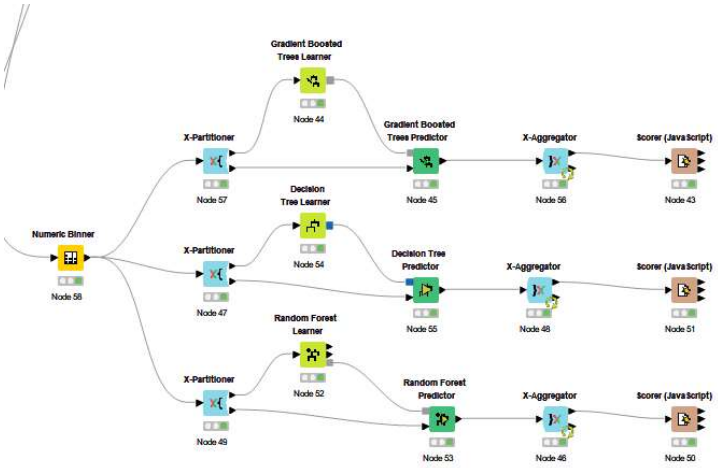


Figura 16- Testes com binning no Knime

Os resultados de cada algoritmo de aprendizagem foram os seguintes:

- Gradient Boosted Tree

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	23378	1342	94.57%
>50K (Actual)	2844	4997	63.73%
	89.15%	78.83%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	23378	2844	4997	1342	94.57%	89.15%	94.57%	63.73%	91.78%
>50K	4997	1342	23378	2844	63.73%	78.83%	63.73%	94.57%	70.48%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
87.14%	12.86%	0.624	28375	4186

Figura 17- Scorer do algoritmo Grandient Booster Tree após binning

- Decision Tree

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	22100	2620	89.40%
>50K (Actual)	3032	4809	61.33%
	87.94%	64.73%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	22100	3032	4809	2620	89.40%	87.94%	89.40%	61.33%	88.66%
>50K	4809	2620	22100	3032	61.33%	64.73%	61.33%	89.40%	62.99%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
82.64%	17.36%	0.517	26909	5652

Figura 18- Scorer do algoritmo Decision Tree após binning

- Random Forest

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	23475	1245	94.96%
>50K (Actual)	3090	4751	60.59%
	88.37%	79.24%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	23475	3090	4751	1245	94.96%	88.37%	94.96%	60.59%	91.55%
>50K	4751	1245	23475	3090	60.59%	79.24%	60.59%	94.96%	68.67%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
86.69%	13.31%	0.604	28226	4335

Figura 19- Scorer do algoritmo Random Forest após Binning

Tendo em conta estes resultados, conseguimos ver que, mais uma vez, o algoritmo de ***Gradient Boosted Tree*** foi o mais eficiente, obtendo uma *accuracy* de 87.14%. Já os algoritmos ***Decision Tree*** e ***Random Forest*** obtiveram *accuracy* de 82.64% e de 86.69%, respetivamente. As precisões não alteraram significativamente. Já as vertentes de *Recall* e *Precision,* continuam baixas, chegando até a baixarem ligeiramente relativamente ao teste de controlo. Devido a tal, não iremos otimizar este método.

2.4.3. Numeric Outliers

Para além do método de Binning, tentamos ainda inserir o nodo **Numeric Outliers** como tratamento adicional de dados, com o intuito de tentar obter melhores resultados. Após algumas tentativas para descobrir qual o dado que nos deveríamos focar para remover os outliers estatísticos, optamos pela coluna de *Capital.*

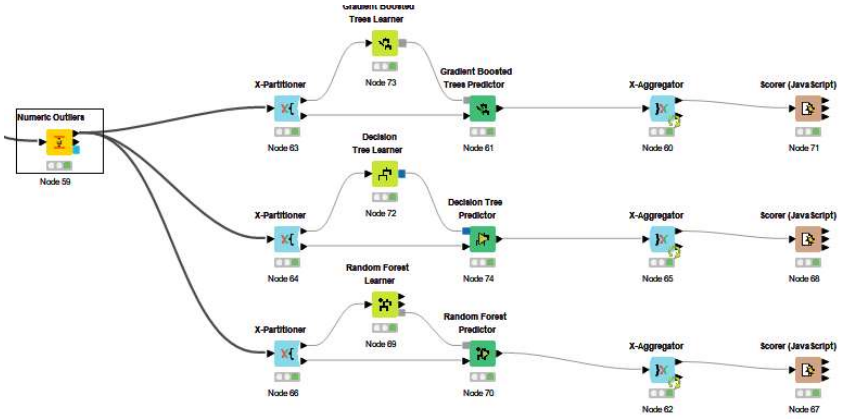


Figura 20- Testes após remoção de outliers no Knime

Eis os resultados para este método:

- Gradient Booster Tree

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	22838	1882	92.39%
>50K (Actual)	3180	4661	59.44%
	87.78%	71.24%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	22838	3180	4661	1882	92.39%	87.78%	92.39%	59.44%	90.02%
>50K	4661	1882	22838	3180	59.44%	71.24%	59.44%	92.39%	64.81%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
84.45%	15.55%	0.549	27499	5062

Figura 21- Scorer do algoritmo Gradient Booster Tree após remoção de Outliers

- Decision Tree

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	21552	3168	87.18%
>50K (Actual)	3476	4365	55.67%
	86.11%	57.95%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	21552	3476	4365	3168	87.18%	86.11%	87.18%	55.67%	86.64%
>50K	4365	3168	21552	3476	55.67%	57.95%	55.67%	87.18%	56.78%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
79.60%	20.40%	0.434	25917	6644

Figura 22- Scorer do algoritmo Decision Tree após remoção de Outliers

- Random Forest

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	23034	1686	93.18%
>50K (Actual)	3517	4324	55.15%
	86.75%	71.95%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	23034	3517	4324	1686	93.18%	86.75%	93.18%	55.15%	89.85%
>50K	4324	1686	23034	3517	55.15%	71.95%	55.15%	93.18%	62.44%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
84.02%	15.98%	0.525	27358	5203

Figura 23- Scorer do algoritmo Random Forest após remoção de outliers

Analisando os resultados obtidos neste método, conseguimos ver que, mais uma vez, o algoritmo de ***Gradient Boosted Tree*** é o mais eficiente, obtendo uma *accuracy* de 84.45%. Já os algoritmos de ***Decision Tree*** e ***Random Forest*** obtiveram *accuracy* de 79.60% e de 84.02%, respetivamente. Mais uma vez, as alterações nas precisões não alteraram positivamente, apesar de que este parâmetro de remoção de outliers foi aquele no qual encontramos mais sucesso.

2.4.4. Clustering

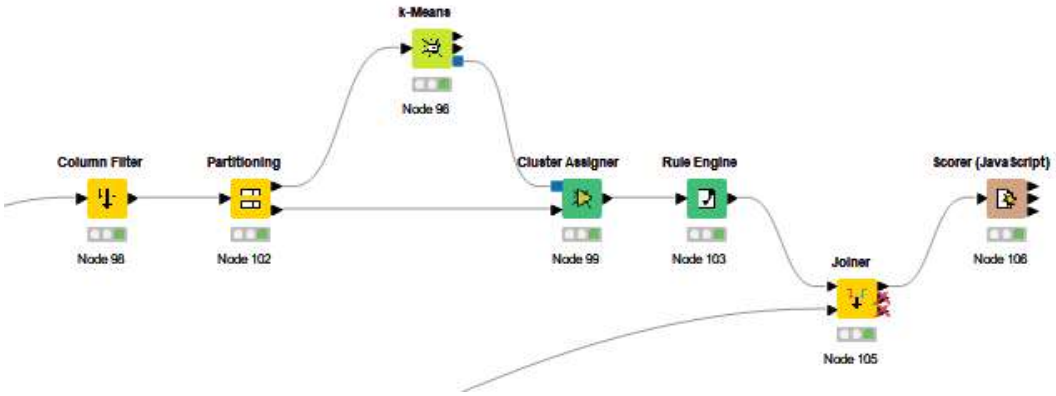


Figura 24- Algoritmo de aprendizagem de Clustering em Knime

Para aplicarmos o método de clustering no nosso modelo, começamos por aplicar um **Column Filter** que removeu a coluna *Income.* Desta forma, tornamos isto num problema de aprendizagem não supervisionado. Depois disso, aplicamos o nodo **K-Means** e o **Cluster Assigner.** Analisando os resultados obtidos, conseguimos ver que o cluster0 correspondia a indivíduos *Income* da classe <=50k. Juntando as colunas originais com esta nova previsão através do nodo **Joiner,** conseguimos obter os seguintes resultados.

Scorer View

Confusion Matrix

	<=50K (Predicted)	>50K (Predicted)	
<=50K (Actual)	5493	1861	74.69%
>50K (Actual)	1808	607	25.13%
	75.24%	24.59%	

Class Statistics

Class	True Positives	False Positives	True Negatives	False Negatives	Recall	Precision	Sensitivity	Specificity	F-measure
<=50K	5493	1808	607	1861	74.69%	75.24%	74.69%	25.13%	74.96%
>50K	607	1861	5493	1808	25.13%	24.59%	25.13%	74.69%	24.86%

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
62.44%	37.56%	-0.002	6100	3669

Figura 25- Scorer do método de Clustering

Como podemos ver, a *accuracy* obtida é de 62.44%, que é bastante reduzida, o que nos leva a acreditar que este método não é uma boa escolha para melhorar os resultados pretendidos.

2.5. Análise dos Resultados

Após realizarmos alguns modelos de aprendizagem no trabalho, conseguimos aferir que sempre que tentávamos aplicar uma nova maneira de organizar os dados, as nossas previsões não melhoravam. O próprio teste base em si não demonstra uma *accuracy* particularmente elevada. Acreditamos que a elevada discrepância na percentagem de indivíduos com classes de salários diferentes torne este dataset em específico difícil de treinar. Não só pelo fator de salário, mas também na discrepância de outros fatores, nomeadamente *age, education, workclass, sex…*

Talvez com a aplicação de um método de *Downsampling* onde as percentagens fossem mais equilibradas, conseguiríamos obter mais resultados. Só que como o número de entradas iria descer consideravelmente, o método em si poderia não ser tão preciso.

3. Previsão da produção de vestuário («*actual_productivity*»)

3.1. Características do *Dataset*

Este *dataset* apresenta 1197 linhas e 13 colunas, as quais representam:

- Como variáveis independentes:
  1. ***date***: Data no formato de DD/MM/YYYY;
  2. ***department***: Departamento associado com a instância;
  3. ***team***: Número da equipa associado com a instância;
  4. ***targeted_productivity***: Produtividade esperada de cada equipa, a cada dia;
  5. ***smv***: Standard Minute Value, ou seja, o tempo alocado para a realização de uma tarefa;
  6. ***wip***: Work in progress, que inclui o número de itens/produtos inacabados;
  7. ***over_time***: Representa o tempo extra dado por cada equipa em minutos;
  8. ***incentive***: Representa a quantidade de incentivo financeiro (em BDT) que possibilita ou motiva uma determinada ação;
  9. ***idle_time***: A quantidade de tempo em que a produção foi interrompida devido a várias razões;
  10. ***idle_men***: O número de trabalhadores que estavam parados devido à interrupção da produção;
  11. ***no_of_workers***: Número de trabalhadores em cada equipa;
  12. ***no_of_style_change***: Número de mudanças no estilo de um produto específico.

- Como variáveis dependentes:
  13. ***actual_productivity***: A real percentagem de produtividade obtida pelos trabalhores.

Das quais temos:

- **Categóricas** -> *date, department, team, wip;*
- **Contínuas** -> *targeted_productivity, smv, over_time, incentive, idle_time, idle_men, no_of_workers, no_of_style_change, actual_productivity*;

3.2. Análise do *Dataset*

3.2.1. Department

Este *dataset* tem no início representados 5 departamentos, os quais, após uma análise mais profunda utilizando um *pie chart*, verificamos que contêm erros ortográficos, existindo apenas 2 corretos (sweing e finishing) e 3 que estão mal escritos, sendo isto visível em:

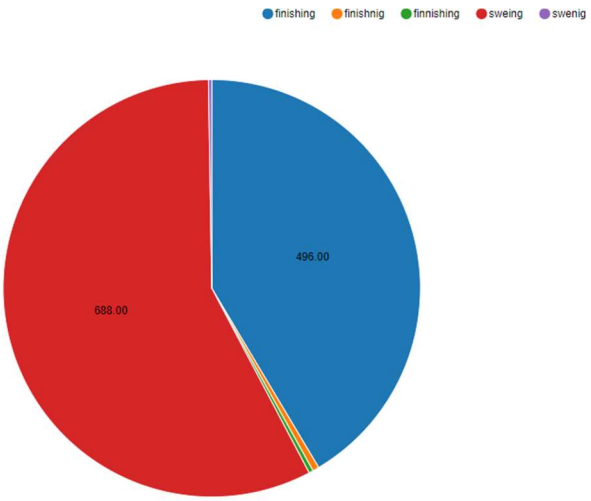


Figura 26- Pie chart do Department

3.2.2. Targeted_Productivity

Este *dataset* possui uma *targeted_productivity*, no geral, positiva, verificando então que, o objetivo final será possuirmos uma *actual_productivity* tão boa, se não melhor, do que esta agora analisada:

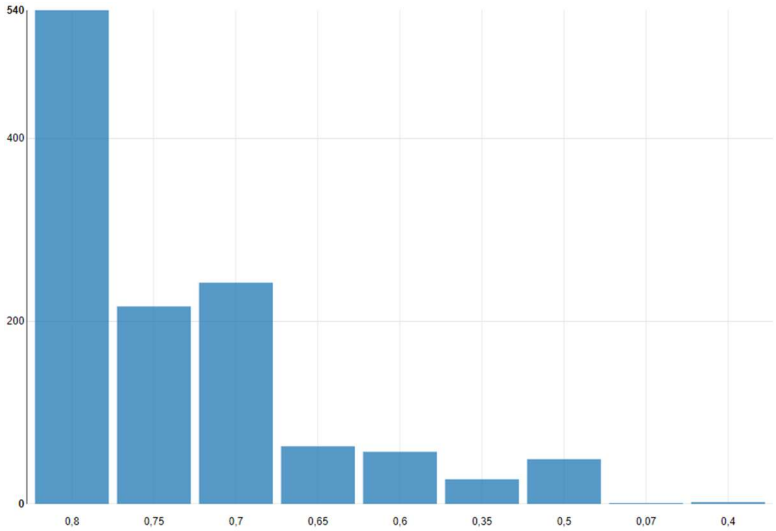


Figura 27- Bar chart da Targeted_Productivity

Possuindo as duas a seguinte correlação, obtida através da análise de um *scatter plot*:

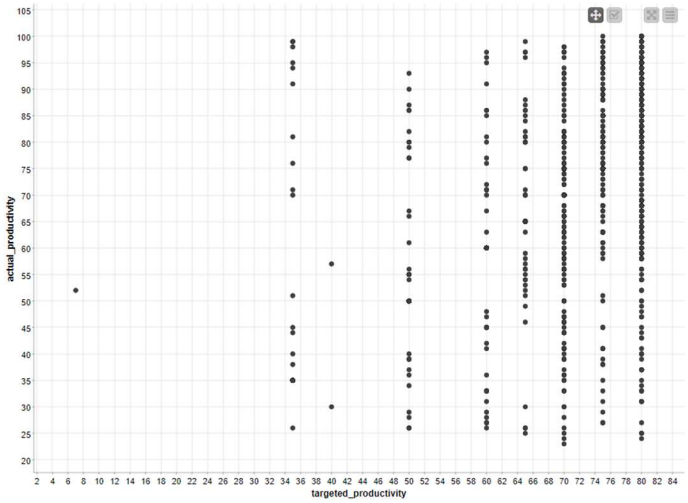


Figura 28-ScatterPlot da actual_productivity e targeted_productivity

3.2.3. SMV

Onde verificamos a presença de *outliers,* ou seja, pontos que se afastam significativamente da maioria dos outros, indicando estes valores extremos ou dados incomuns, estando também presente uma densidade de pontos mais marcante quando o SMV é menor (esquerda) e mais ou menos constante no resto (meio do *scatter plot*, sem contar com os *outliers*).

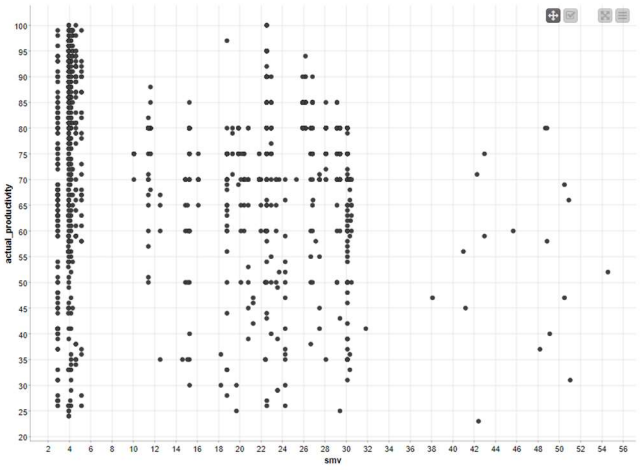


Figura 29- ScatterPlot do SMV e da actual_productivity

Verificando-se também mais *outliers* em outros parâmetros, como é o caso de:

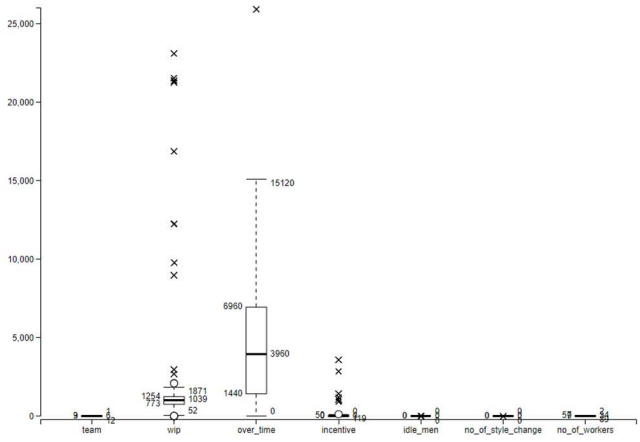


Figura 30- Boxplot do Dataset

3.2.4. Wip

A distribuição dos dados de *work in progress é evidenciada pelo seguinte histograma.*

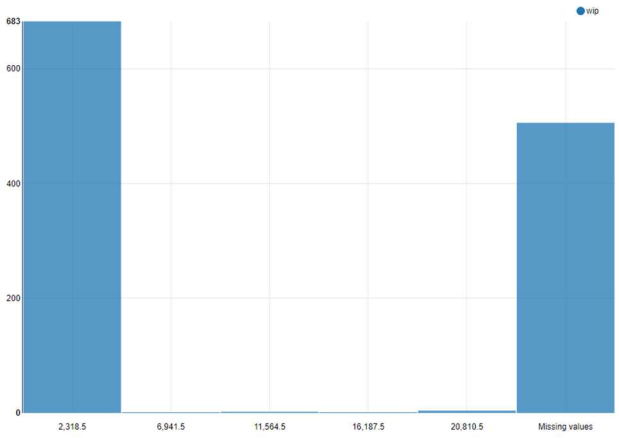


Figura 31- Bar Chart do WIP

Onde é evidente que a maior parte das instâncias foram feitas com um *WIP* contido no primeiro intervalo representado, seguindo-se pela imensa presença de "*NaN*" (*missing value*), representado pelo último intervalo visível em cima.

3.2.5. Idle_time

Quer o *Idle_time* como o *Idle_men* (mencionado anteriormente) têm em comum a característica de possuírem uma grande dimensão de valores iguais a 0, o que é nos possível observar através de:

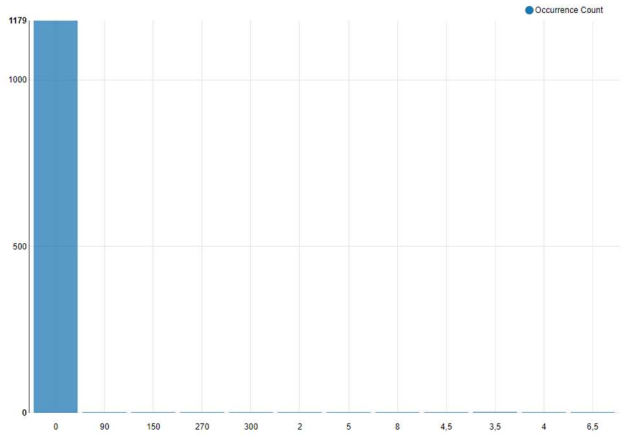


Figura 32- Bar Chart do idle_time

idle_men	☐	0	45	0.369	3.269	10.686	9.855

Figura 33-Linha do nodo Data Explorer correspondente ao idle_men

3.3. Pré-Processamento de Dados

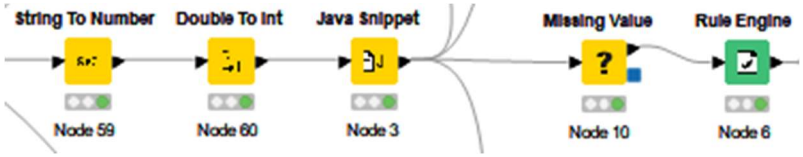


Figura 34-Nodos de Pré-Processamento no Knime

Começamos pelo tratamento do campo *no_of_workers*, utilizando os dois primeiros nodos do *workflow* para converter esse campo de S*tring* para *Integer*, fazendo-o assim assumir valores inteiros e não decimais. Logo após isto, visamos tratar do WIP, substituindo, através do *Java Snippet*, os valores de *NaN* por *Missing Values*, que são posteriormente tratados para assumir o valor da média, seguindo-se então a abordagem mais comum e simples para o tratamento de *Missing Values*. Visto isto, aplicamos de seguida o tratamento necessário à coluna relativa aos departamentos, através do *Rule Engine*.

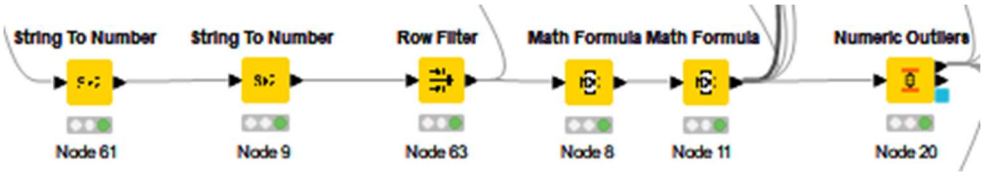


Figura 35- Nodos de Pré-Processamento no Knime 2

Seguindo-se agora o tratamento do resto dos dados numéricos lidos como *Strings*, através da conversão do *SMV, targeted_productivity, actual_productivity e idle_time* para *Double*, a partir do qual aproveitamos para filtrar pela *actual_productivity*, visando através disto garantir que os seus valores se encontram entre 0 e 1.

Com isto feito, e para melhorar a nossa percetibilidade dos dados, decidimos passar ambas as produtividades registadas para intervalos maiores, indo assim de [0,1] para [0,100] através do nodo *Math Formula*.

Face a isto, e procurando sempre o melhor tratamento dos dados possíveis, focamo-nos agora no tratamento dos *outliers* observados anteriormente, tratamento este que, após uma nova análise dos dados com o *data explorer*, verificamos que melhorou os mesmos e nos deixou com nenhum caso de *Missing Values*, não sendo preciso o tratamento destes.

3.4. Modelação

Com todos os dados tratados, passamos agora para a fase de modelação, onde utilizamos, não só conceitos de Redes Neurais Artificiais (*RNAs*) dadas nas aulas, visível em:

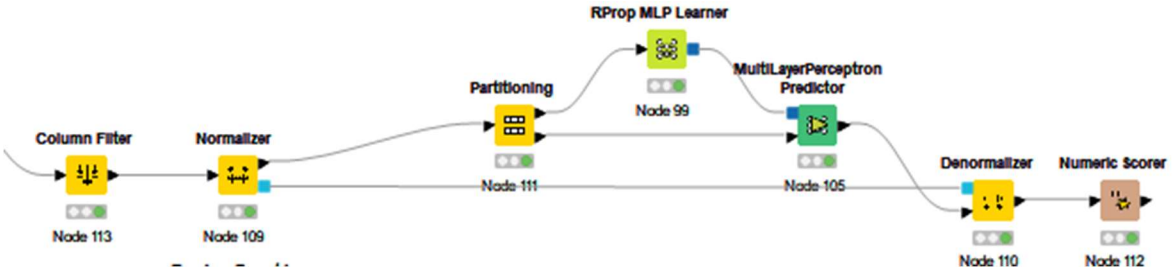


Figura 36- Modelo de RNA em Knime

Onde obtemos o seguinte resultado:

R²:	-16,623
Mean absolute error:	71,227
Mean squared error:	5 375,163
Root mean squared error:	73,316
Mean signed difference:	-71,227
Mean absolute percentage error:	0,991
Adjusted R²:	-16,623

Figura 37- Resultado obtido no Numeric Scorer

Como também utilizamos *metanodos* que estão a correr durante algum tempo e que visão descobrir as melhores combinações de variáveis possíveis e escolher a filtragem por nós pretendida, ou seja, encontram-se a fazer o processo de *feature selection*, juntamente com nodos de *cross-validation* designados por *X-Partitioner e X-Aggregator*, que permitem a utilização de uma maior quantidade de dados.

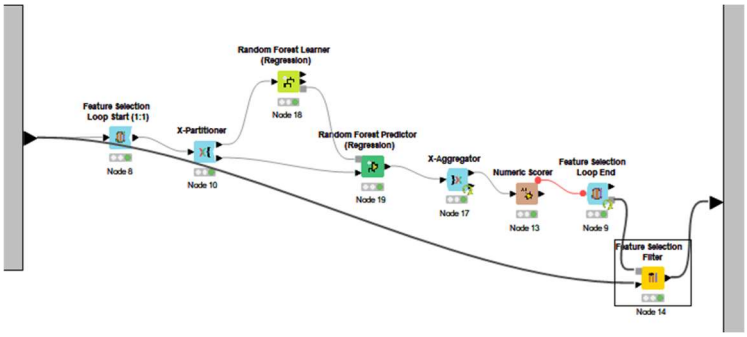
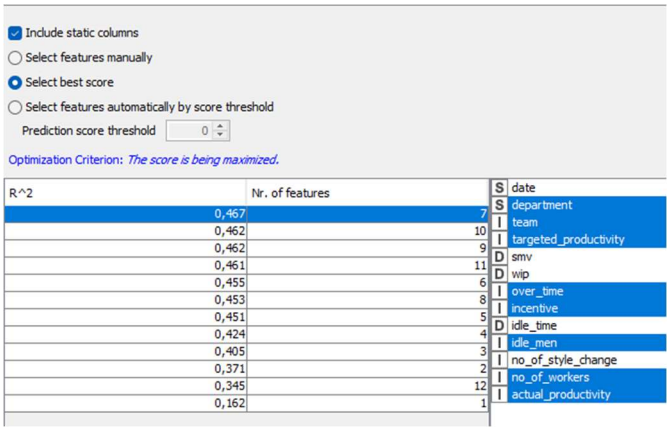


Figura 38- Conjunto de Metanodos utilizado



R^2	Nr. of features
0,467	7
0,462	10
0,462	9
0,461	11
0,455	6
0,453	8
0,451	5
0,424	4
0,405	3
0,371	2
0,345	12
0,162	1

S	date
S	department
I	team
I	targeted_productivity
D	smv
D	wip
I	over_time
I	incentive
D	idle_time
I	idle_men
I	no_of_style_change
I	no_of_workers
I	actual_productivity

Figura 39- Especificações da filtração

Onde realizamos os seguintes estudos, sendo importante mencionar que o *X-Partitioner* é configurado para utilizar 10 validações e realizar *Random Sampling* em todos os instâncias em que é utilizado:

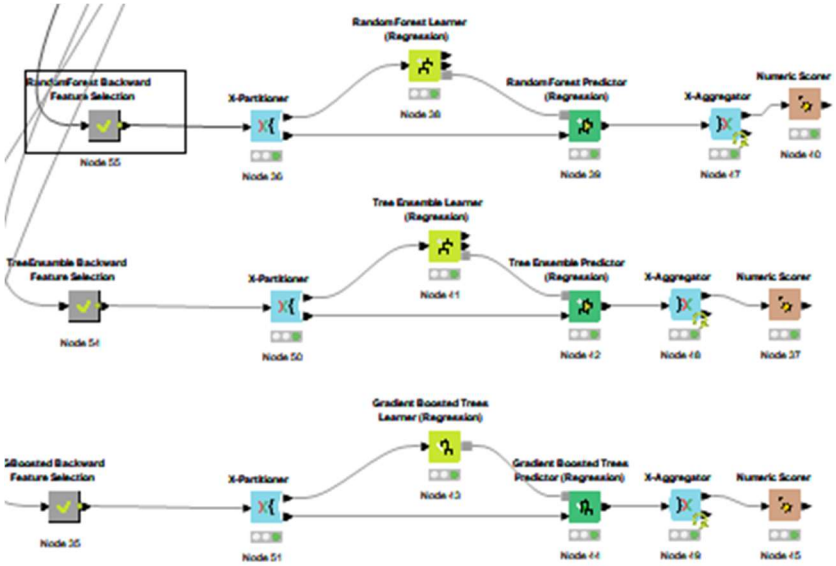


Figura 40- Modelo realizado para o problema de Regressão

Que nos trazem como resultados, respetivamente:

R²:	0,455
Mean absolute error:	8,081
Mean squared error:	156,066
Root mean squared error:	12,493
Mean signed difference:	0,051
Mean absolute percentage error:	0,149
Adjusted R²:	0,455

R²:	0,456
Mean absolute error:	8,069
Mean squared error:	155,914
Root mean squared error:	12,487
Mean signed difference:	0,109
Mean absolute percentage error:	0,149
Adjusted R²:	0,456

R²:	0,457
Mean absolute error:	7,88
Mean squared error:	155,46
Root mean squared error:	12,468
Mean signed difference:	0,807
Mean absolute percentage error:	0,146
Adjusted R²:	0,457

Figura 41- Resultados obtidos

Após isto, e na esperança de melhorar os nossos resultados, decidimos implementar a técnica de *UpSampling*, utilizando um *Auto-Binner*, com as seguintes características:

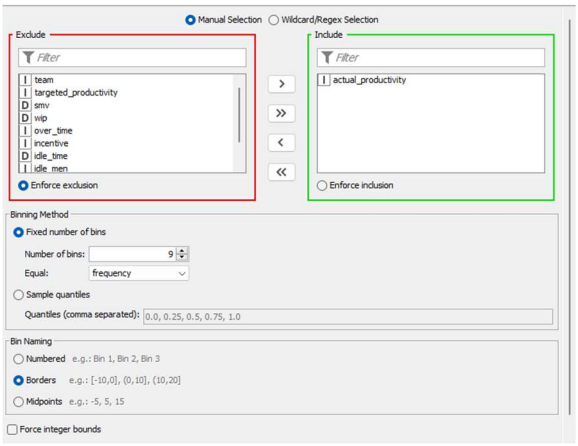


Figura 42- Especificações do Auto-Binner

E do qual resulta este novo estudo, que nos fornece os seguintes resultados:

Figura 43- Modelo utilizado para o problema de Classificação

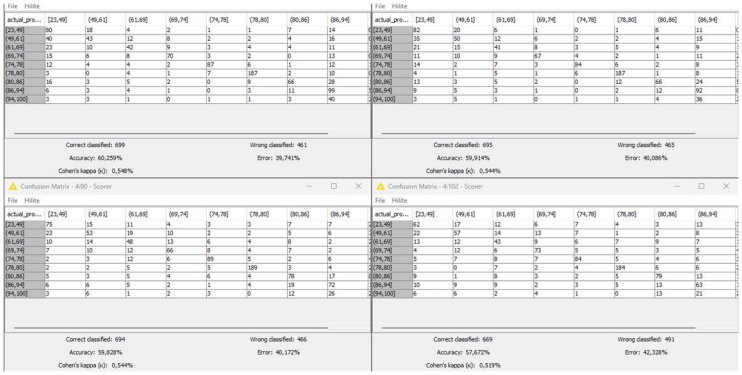


Figura 44- Resultados Obtidos

3.5. Análise dos Resultados

Desta forma, consideramos que este *dataset* continha uma quantidade de dados não muito proveitosa para efeitos de aprendizagem, embora tenha apenas necessitado de tratamentos relativamente simples, visto termos obtido um resultado, em geral, não muito positivo.

Concluindo este trabalho por dizer que, com este *dataset*, tratar de um problema de regressão levou-nos a obter os melhores resultados através da *Gradient Boosted Tree* (*R^2: 0.457*). Mas, por sua vez, se este for convertido para um problema de classificação, obtemos melhores resultados com o *Random Forest Learner* (*Accuracy: 60.259%*).