TRABALHO 3

Karla Fátima Calvoso Simões; Weld Lucas Cunha

QUESTÃO 1: Inspecionar os dados de treinamento. Quantos exemplos há de cada classe? O dataset está desbalanceado ?

Há 1007 amostras no conjunto de treinamento e 252 no conjunto de validação. São 16 colunas: 15 colunas de características (numéricas e categóricas) e 1 coluna com o valor objetivo (label). O dataset encontra-se bastante desbalanceado, portanto, foi realizado o oversampling no dataset de treinamento sobre as duas classes menos representadas: dead (x10) e recovered (x6). A quantidade de exemplos em cada classe é apresentada na tabela a seguir:

Classe	Treinamento	Treinamento (ajustado)	Validação
dead	47	470	10
onTreatment	845	845	220
recovered	115	690	22

QUESTÃO 2: Treinar uma árvore de decisão como baseline.

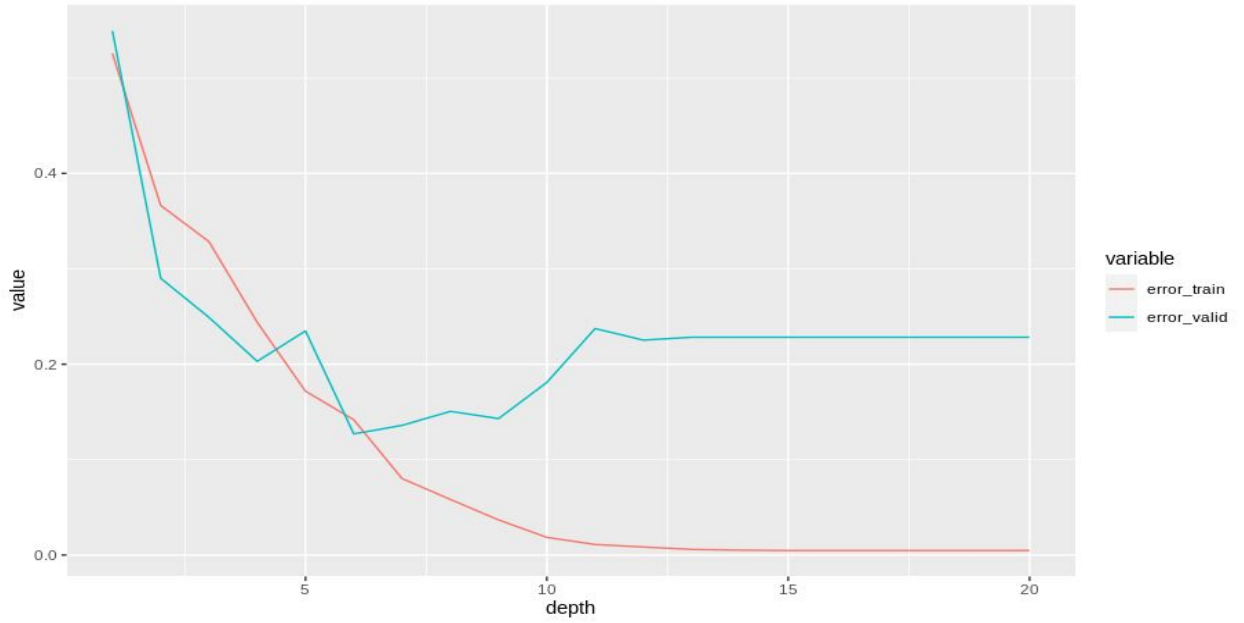
Foi treinado um modelo simples, como baseline, sem limitação do tamanho da árvore, e outro modelo como a versão podada da baseline, os valores de acurácia normalizada são mostrados a seguir para os dois modelos:

Acurácia (norm.)	Baseline	Baseline pruned
Treino	0.9953	0.9953
Validação	0.7717	0.7717

QUESTÃO 3: Treine outras árvores de decisão variando o tamanho das árvores geradas. Plote o erro no conjunto de treinamento e validação pela profundidade da árvore de decisão.

Foram consideradas árvores com valores de profundidade indo de 1 até 20. Os valores

de erro, calculado como: erro = 1 - $Acc_{normalizada}$, são mostrados a seguir:



Observa-se que o modelo de melhor performance apresenta profundidade igual a 6, antes de apresentar características de overfitting.

QUESTÃO 4: Explore pelo menos 2 possíveis subconjuntos de features para treinar uma árvore de decisão. Reporte o erro no treino, validação e teste.

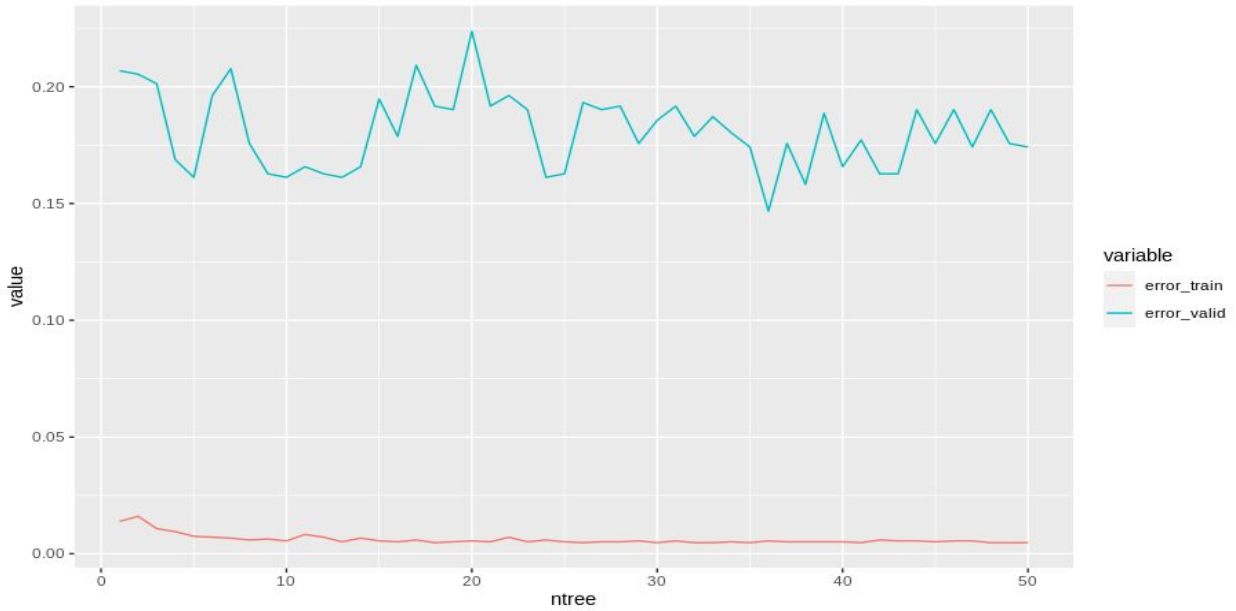
Nesta etapa, foram implementados modelos alternativos com subconjuntos de features, as features consideradas são mostradas na tabela a seguir, assim como os valores de erro apresentados.

---	**Modelo 1**	**Modelo 2**
Features	case_in_country + reporting.date + age	case_in_country + reporting.date + age + international_traveler + domestic_traveler + exposure_start + exposure_end + traveler
Erro (treino)	0.1196	0.0725
Erro (valid.)	0.1974	0.2640
Erro (teste)	0.2983	0.2628

QUESTÃO 5: Treine várias florestas aleatórias variando o número de árvores. Plote o erro no

conjunto e treinamento e validação variando o número de árvores geradas.

Foi realizada uma varredura pelo número de árvores, de 1 a 30, conforme mostrado no gráfico a seguir, com seus respectivos valores de erro de treinamento e validação.



Observa-se que o modelo de melhor performance apresenta número de árvores igual a 36, antes de apresentar características de overfitting.

QUESTÃO 6: Calcule a matriz de confusão, os verdadeiros positivos para cada classe e a acurácia normalizada no teste para os melhores modelos (árvore com melhor profundidade, floresta com melhor número de árvores e árvore com melhor subconjunto de features).

Resumo das performances dos quatro melhores modelos, por categoria:

---	**Baseline (1)**	**Árvore com melhor profundidade (2)**	**Árvore com melhor subconjunto de features (3)**	**Floresta com melhor número de árvores (4)**
Configuração	Árvore simples	Árvore com profundidade 6	Features: case_in_country + reporting.date + age	Floresta com 36 árvores
True positive rate	dead: 0.5833 on treatment: 0.9387 recovered:0.6190	dead: 0.3182 on treatment: 0.9409 recovered: 0.5178	dead: 0.2917 on treatment: 0.9486 recovered: 0.4026	dead: 0.5625 on treatment: 0.9641 recovered: 0.6875
Acc (teste)	0.7122	0.71	0.7017	0.8195

Matrizes de confusão 1:

Pred. \ Ref.	dead	onTreatment	recovered
dead	7	3	2
onTreatment	5	245	11
recovered	1	15	26

Matrizes de confusão 2:

Pred. \ Ref.	dead	onTreatment	recovered
dead	7	14	1
onTreatment	5	223	9
recovered	1	26	29

Matrizes de confusão 3:

Pred. \ Ref.	dead	onTreatment	recovered
dead	7	15	2
onTreatment	5	203	6
recovered	1	45	31

Matrizes de confusão 4:

Pred. \ Ref.	dead	onTreatment	recovered
dead	9	7	0
onTreatment	3	242	6
recovered	1	14	33

CONCLUSÃO

O objetivo deste trabalho é desenvolver um modelo para prever o possível estado de um paciente diagnosticado com COVID-19, onde analisando diversas features devemos inferir se o paciente ficará recuperado, permanecerá em tratamento ou irá falecer.

Etapa 1: Get data: Carregamento dos dois arquivos e divisão entre treinamento e

validação..

Etapa 2: Clean, Prepare and Manipulate Data: Nesta etapa foi realizado o re-balanceamento das classes, mantendo ainda a relação entre a frequência das classes mas diminuindo o desbalanceamento presente nos dados de treinamento. Na codificação das features foi utilizada a técnica de one-hot encoding para o sexo. No caso do país, optamos por utilizar o IDH dos países ao invés de aplicar a técnica de one-hot encoding. Justificativa: Desta forma foi possível codificar o nome do país diretamente em uma grandeza numérica que está diretamente relacionada às condições de saúde, qualidade de vida, educação e renda (Fonte: https://pt.wikipedia.org/wiki/Lista_de_pa%C3%ADses_por_%C3%8Dndice_de_Desenvolvimento_Humano).

Etapa 3: Train Models: Foi realizada implementação de um modelo de árvore como base; modelos de árvore com valor de profundidade ajustado; 2 modelos de árvores com base nos subconjuntos de features e modelos de florestas aleatórios com valores de números de árvores definidos. Nesta etapa foram escolhidos os modelos com melhores performances na validação como melhores de cada categoria.

Etapa 4: Test Data: Nesta etapa, deu a lógica, o modelo de floresta aleatória, sendo o mais poderoso, apresentou o melhor resultado nos testes, tendo aproximadamente 10% a mais de acurácia em relação a todos os outros melhores modelos, os quais tiveram resultados bastante semelhantes.

Etapa 5: Improve: Durante a execução do trabalho, algumas observações foram tomadas em relação à construção dos modelos e como melhorá-los:

- Baseline: Optou-se por utilizar um modelo com todas as features, a sua versão podada apresentou os mesmo resultados.
- Árvore com profundidade máxima ajustada:Neste caso, observa-se que árvores com baixa profundidade são incapazes de descrever os dados, enquanto árvores muito profundas tendem a apresentar overfitting.
- Modelo baseado no subconjunto de features: Neste caso, optamos por dois subconjuntos: o primeiro deles com features relacionados ao número de casos no país, data do relatório do paciente e idade, no segundo, adicionamos informações relacionadas a possibilidade do paciente ter viajado.
- Árvores aleatórias: Neste caso, foi necessário ajustar a quantidade de features que cada árvore aleatória deveria possuir. Modelos com muito poucas features são muito simples e não apresentariam uma boa performance, modelos com muitas features acabam perdendo a possibilidade de ter muitas variações de modelos aleatórios. Optamos por um modelo com 12 features por árvore aleatória.