

# Predição na Base de Dados SpaceShip Titanic do Kaggle

1<sup>st</sup> Halliday Gauss Costa dos Santos  
dept. DECOM  
Universidade Federal de Ouro Preto  
Ouro Preto, Brasil  
halliday.santos@aluno.ufop.edu.br

2<sup>nd</sup> William Gomes de Lima Natividade  
dept. DECOM  
Universidade Federal de Ouro Preto  
Ouro Preto, Brasil  
william.natividade@aluno.ufop.edu.br

**Resumo**—Competições são de grande importância no aprendizado por trazer aos competidores a curiosidade e o desejo para resolver um problema. Isso se mantém área de ciência de dados, em destaque na plataforma Kaggle, que abriga inúmeras competições dessa área. Este trabalho apresenta uma predição na base de dados rotulada SpaceShip Titanic do Kaggle. São utilizadas técnicas de mineração de dados para realizar o descobrimento de informações e as transformações dos dados com ênfase na utilização do algoritmo KNNImputer para o preenchimento de dados ausentes, e posteriormente a aplicação da técnica de *deep learning* TabNet para a classificação.

**Palavras-chave**—Kaggle. SpaceShip Titanic. Mineração de Dados. Valores Ausentes. KNNImputer. Aprendizado Profundo. TabNet. Predição.

**Abstract**—Competitions are of great importance in learning because they bring to the contestants the curiosity and desire to solve a problem. This remains an area of data science, highlighted in the Kaggle platform that hosts numerous competitions in this area. This work presents a prediction in Kaggle's SpaceShip Titanic database. Data mining techniques are used to discover information and transform data with emphasis on the use of the KNN Imputer algorithm to fill in missing data, and later the application of the TabNet deep learning technique for classification.

**Index Terms**—Kaggle. SpaceShip Titanic. Data Mining. Missing Values. KNNImputer. Deep Learning. TabNet. Prediction.

## I. INTRODUÇÃO

As competições trazem consigo um papel importante para o aprendizado e melhoria pessoal pois instigam a curiosidade dos competidores. Isso se mantém mesmo nos desafios voltados para a área da computação, por exemplo, predições sobre uma base de dados, além de que disputas em problemas de classificação motivam as pessoas que atuam na área de ciência de dados a conseguir um resultado superior ao melhor existente.

O Kaggle [1] é uma plataforma online de ciência de dados que hospeda competições, além de abrigar uma grande comunidade de competidores e cientistas de dados de diversas origens. Cada competidor pode participar das competições em aberto, submeter seus resultados na plataforma, e em seguida verificar o desempenho obtido e a classificação no ranking da competição. Além disso, o Kaggle possui fóruns de discussão onde são compartilhados códigos e ideias possibilitando o aprendizado tanto de iniciantes quanto de cientistas da área.

Muitas competições tem fins acadêmicos ou são de interesse privado, e algumas valem recompensas [2].

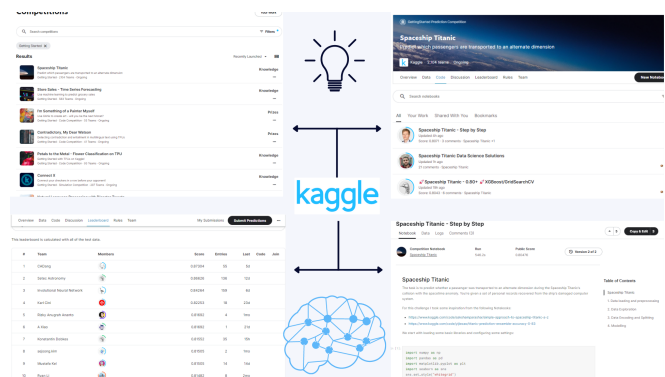


Fig. 1. Funcionalidades do Kaggle.

Para resolver esses problemas, de classificação ou de regressão, por exemplo, os cientistas de dados utilizam de diversas técnicas e algoritmos [3], alguns populares são:

- Regressão Logística: um dos métodos mais populares usados para classificar dados binários e realizar regressões. Este método baseia-se que o rótulo (variável dependente) pode ser expressado em função das variáveis independentes (atributos), levando à função de regressão. Essa função recebe como entrada as variáveis independentes e retorna a predição.
- Árvore de Decisão: são algoritmos simples e muito utilizados. Uma árvore de decisão é um modelo estruturado em árvore com nós de decisão e nós de previsão (nós folhas). Os nós de decisão são usados para ramificar e os nós de previsão especificam rótulos da classe. Geralmente essas árvores são construídas utilizando a abordagem dividir para conquistar e baseia-se no ganho de informação de cada atributo.
- Random Forest: algoritmo de classificação que utiliza um conjunto de árvores de decisão para realizar a predição. É um dos algoritmos de aprendizado mais usados e precisos. Cada árvore que compõe a floresta é construída através de divisões feitas na base de teste baseadas na pureza do subconjunto. Durante a classificação cada

árvore irá prever um rótulo e a classe mais popular é retornada.

- Naive Bayes: um algoritmo de aprendizado indutivo eficaz. O algoritmo é baseado no Teorema de Bayes, assume que o efeito do valor de um atributo sobre uma determinada classe é independente dos valores dos demais atributos. Esta é a suposição de independência condicional e verdadeira em aplicações do mundo real. Devido a isso, o NB funciona bem em conjuntos de dados complexos e de alta dimensão.
- KNN: o KNN é um algoritmo de classificação utilizado quando há pouco ou nenhum conhecimento prévio sobre a distribuição dos dados. Usando as métricas de distância para medir a proximidade entre as amostras de treinamento e a amostra de teste, KNN atribui à amostra de teste a classe de suas  $k$  amostras de treinamento mais próximas. Existem várias medidas de distâncias que podem ser utilizadas para implementar esse classificador, por exemplo: a Distância Euclidiana e a Distância de Manhattan. Em algumas implementações se as instâncias possuem algum atributo nominal, na fórmula da distância a diferença entre os valores será igual a 1 se os valores dos atributos forem diferentes e 0 em caso contrário.
- Redes Neurais: as redes neurais são um conjunto de unidades de entrada e saída conectadas, sendo que cada conexão tem um peso associado. A rede aprende por ajustar os pesos para ser capaz de prever as classes das tuplas de entrada. Uma rede neural tradicional consiste em uma camada de entrada, uma ou mais camadas ocultas com vários neurônios, e uma camada de saída (camada de classificação). O Multilayer Perceptron (MLP) é um tipo de RNA que tem capacidade de resolver problemas de classificação não linear com alta precisão e bom desempenho de generalização. O MLP usa retropropagação para propagar o erro na direção inversa para ajustar os pesos.

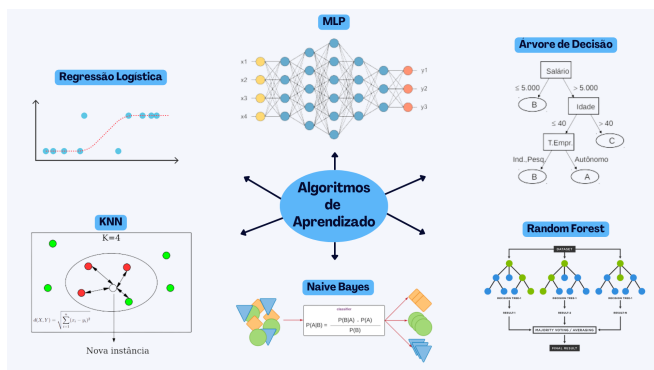


Fig. 2. Alguns algoritmos de aprendizado.

Apesar de existirem bons métodos e algoritmos para realizar uma tarefa de classificação, a qualidade dos dados impactam diretamente no desempenho desses classificadores e é uma

das principais preocupações no Aprendizado de Máquina. Portanto, a etapa de pré-processamento de dados deve ser feita com cuidado, principalmente no que diz respeito ao preenchimento de valores ausentes na base de treinamento e de teste, já que muitos algoritmos de aprendizado não funcionam com a presença de valores nulos. Dessa forma, o tratamento de valores ausentes deve ser cuidadosamente planejado, caso contrário, distorções podem ser introduzidas no conhecimento e o desempenho dos classificadores pode ser prejudicado [4].

Uma das formas mais comuns para lidar com dados ausentes é preencher os valores nulos com a média ou com a moda, também é muito utilizado a imputação de valores ausentes utilizando métodos de regressões. Esses métodos funcionam muito bem quando o conjunto de dados é formado apenas por atributos contínuos, no entanto, o cenário muda quando há a presença de atributos nominais [5].

Armina [6] apresenta uma descrição dos algoritmos mais utilizados para a imputação de um valor ausente no conjunto de dados, dentre eles o KNNImputer (*K Nearest Neighbor Imputer*), o qual segue uma abordagem local, ou seja, explora a estrutura de similaridade local nos conjuntos de dados, portanto, somente o subconjunto dos dados que possuem altas correlações com os atributos faltantes será usado pelo algoritmo. O KNNImputer consiste em gerar os  $K$  vizinhos mais próximos de uma instância utilizando o algoritmo KNN e em seguida ajuda a prever os valores nulos da instância em questão baseando-se nas instâncias mais próximas.

Apesar do sucesso no uso das Redes Neurais Profundas (DNN) para resolução de problemas envolvendo textos, imagens e áudios, na maioria das vezes elas não possuem bons resultados em problemas de classificação envolvendo dados tabulares comparado as diversas técnicas atuais de aprendizado, como, por exemplo, as técnicas supracitadas [7]. Para resolver esse problema foi criada a DNN TabNet, a qual possui as seguintes características:

- Flexibilidade: é possível inserir dados tabulares brutos sem qualquer pré-processamento e treinar utilizando otimização baseada em gradiente descendente.
- Capacidade de Aprendizado: o TabNet usa atenção sequencial para escolher quais características serão utilizadas em cada etapa de decisão, permitindo interpretabilidade e melhor aprendizado ao destacar as características mais relevantes. A seleção de características é por instância (Figura 3).
- Desempenho: supera ou está no mesmo nível de outros modelos de aprendizado tabular em vários conjuntos de dados para problemas de classificação e regressão de diferentes domínios.
- Pré-treinamento não supervisionado: é possível utilizar pré-treinamento para aprender relações e prever os valores nulos ou mascarados. Os pesos aprendidos podem então ser salvos e usados para uma tarefa supervisionada. O uso desse recurso aumenta muito o desempenho da tarefa proposta (Figura 4).

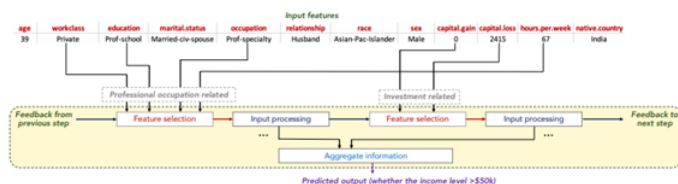


Fig. 3. Procedimento de seleção de características da TabNet.

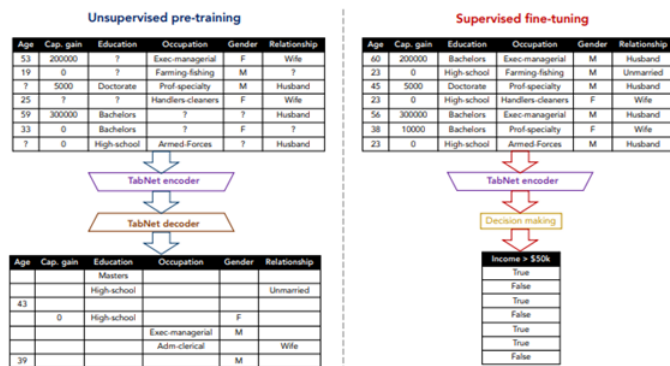


Fig. 4. Aprendizado tabular auto-supervisionado da TabNet.

O objetivo desse trabalho é realizar uma predição sobre a base de dados do Kaggle chamada: SpaceShip Titanic, utilizando a DDN TabNet. Também serão utilizadas técnicas de mineração de dados para: o descobrimento de informações, através da análise dos atributos, a realização das transformações dos dados, e o pré-processamento com ênfase no uso do KNNImputer no preenchimento dos valores nulos.

## II. TRABALHOS RELACIONADOS

Em seu trabalho, Kothar Yash [8] utiliza a base de dados Titanic disponível no Kaggle, um popular site de ciência de dados. Tal base reúne informações sobre cada passageiro presente no Titanic, incluindo 11 variáveis descritivas associadas a 891 passageiros no conjunto de treino e 418 no conjunto de teste. De antemão o autor apresenta uma visão geral e descrição do problema abordado, analisando os atributos disponíveis e discutindo correlações identificadas, também foi analisada a importância de cada variável para se alcançar o objetivo: prever quais passageiros sobrevivem e quais morrem. A seguir ele documenta sua metodologia utilizada para pré-processamento dos dados das bases de dados (treino e teste), envolvendo remoção de variáveis irrelevantes e transformação das relevantes. Neste estudo utilizou-se os algoritmos Naive Bayes, Decision Tree e Random Forest; submentendo os resultados na plataforma Kaggle concluiu-se que o algoritmo Naive Bayes obteve melhor resultado que os demais com 92,52% de acurácia. Esta classificação estimou os parâmetros necessários usando probabilidade Gaussiana simples e equações de distribuição para normalizar os dados.

O objetivo do trabalho de [9] é propor algoritmos de Aprendizado de Máquina para prever mecanismos de ação (MoA), ou seja, a atividade biológica de uma molécula, utilizando a

base de dados *Mechanisms of Action (MoA) Prediction* do Kaggle. A seleção dos modelos foi realizada por meio de um concurso público online que elegeu os modelos com melhor desempenho para o conjunto de dados fornecido. A acurácia das soluções é avaliada pelo valor médio da função de perda logarítmica aplicada a cada par de anotações de droga-MoA.

Nesse trabalho foi feito a remoção dos atributos que não geram nenhum mecanismo de ação, a aplicação do Quantile Transformer no intuito de espalhar os valores mais frequentes e diminuir o impacto de valores discrepantes, a aplicação PCA, por fim foi acrescentado novos atributos baseando-se na soma, média, desvio padrão, curtose e assimetria sobre as características de expressão gênica e viabilidade celular.

Para validação foi utilizada a Estratificação Multilabel, que divide o conjunto de dados em vários conjuntos, distribuindo os alvos positivos em todos os conjuntos da maneira mais uniforme possível. Cada conjunto não usado para o treinamento é utilizado para avaliar a pontuação de validação cruzada, a qual é dada pela *log-loss* (entropia cruzada).

Os modelos utilizados foram:

- NN: composta por uma camada de entrada, uma camada oculta com 1000 neurônios e uma camada de saída, com a inserção de um *dropout* após cada camada e o uso da função de ativação de unidade linear retificada com vazamento (LeakyReLU), otimizador Adam e função de perda *log-loss*.
- DNN TabNet: uma largura de 32 para a camada de previsão de decisão e uma largura de 32 para a incorporação de atenção para cada máscara, 1 na etapa da arquitetura, valor de gama de 1.3, otimizador Adam com uma taxa de aprendizado de  $2e-2$  e um decaimento de peso de  $1e-5$ , um coeficiente de perda de esparsidade de 0 e entmax como função de mascaramento. Treinado com um tamanho de lote de 1024 e um tamanho de lote virtual de 128 por 200 épocas antes de ser interrompido por um parâmetro de paciência de 50 épocas.
- ResNet: primeiramente, o modelo é treinado para prever alvos não pontuados com o otimizador Adam por 50 épocas, minimizando a perda de entropia cruzada binária, com uma taxa de aprendizado de  $1e-3$  para 128 de tamanho de lote. O escalonador de taxa de aprendizado ReduceLROnPlateau é usado com um fator de 0.5 e uma paciência de 4 épocas. Em seguida, é utilizado a transferência de aprendizado desse modelo para uma outra ResNet que é treinada da mesma forma que a NN supracitada.

Os resultados obtidos pela NN, Tabnet e ResNet são 0.0159, 0.0150 e 0.0147, respectivamente no conjunto de validação. Ao fazer um ensemble desses 3 modelos, que produz a média ponderada das saídas, e submeter ao kaggle, foi possível obter as seguintes pontuações: 0.01626 (NN), 0.01633 (TabNet), 0.01644 (ResNet) 0.1610 (Ensemble), sendo o Ensemble o melhor dos modelos, o qual garantiu a posição 72º no ranking.

### III. BASE DE DADOS

Nessa seção será apresentado o problema e as características encontradas na base de dados Spaceship Titanic do Kaggle através de técnicas de mineração de dados.

#### A. O problema

No ano de 2912, foi recebido uma transmissão de quatro anos-luz de distância. A nave espacial Titanic com quase 13.000 passageiros a bordo, partiu em uma viagem inaugural transportando emigrantes do nosso sistema solar para três exoplanetas recém-habitáveis orbitando estrelas próximas. Durante o percurso a nave colidiu com uma anomalia do espaço-tempo escondida dentro de uma nuvem de poeira. Embora a nave tenha permanecido intacta, quase metade dos passageiros foi transportada para uma dimensão alternativa. O problema envolve ajudar a prever quais passageiros foram transportados pela anomalia através de uma base de dados com registros recuperados do sistema de computador danificado.

#### B. Descrição dos dados

O problema conta com duas bases de dados: uma para a realização do treinamento com 8693 registros e 14 atributos, e a outra, que deve ser utilizada para o teste na plataforma, possuindo 4277 registros e 13 atributos.

Antes de aplicar técnicas de limpeza e transformações de dados é necessário entender quais são os dados que a base de dados possui. A tabela I abaixo mostra a descrição completa dos dados.

Atributo	Descrição
PassengerId	Um ID único para cada passageiro. Cada ID indica um grupo com o qual o passageiro está viajando e seu número dentro do grupo. As pessoas em um grupo geralmente são membros da família, mas nem sempre.
HomePlanet	O planeta de onde o passageiro partiu, normalmente seu planeta de residência permanente.
CryoSleep	Indica se o passageiro optou por ser colocado em animação suspensa durante a viagem. Os passageiros em sono criogênico estão confinados em suas cabines.
Cabin	O número da cabine onde o passageiro está hospedado. No formato deck/num/side, onde side pode ser P para Bombordo ou S para Estibordo.
Destination	O planeta para o qual o passageiro irá desembarcar.
Age	A idade do passageiro.
VIP	Se o passageiro pagou pelo serviço VIP especial durante a viagem.
RoomService	Valor que o passageiro gastou na comodidade de luxo: Serviço de Quarto.
FoodCourt	Valor que o passageiro gastou na comodidade de luxo: Praça de Alimentação.
ShoppingMall	Valor que o passageiro gastou na comodidade de luxo: Shopping.
Spa	Valor que o passageiro gastou na comodidade de luxo: Spa.
VRDeck	Valor que o passageiro gastou na comodidade de luxo: Área de Jogos.
Name	O nome e sobrenome do passageiro.
Transported	Se o passageiro foi transportado ou não para outra dimensão. Rótulo que deve ser predito.

TABLE I  
DESCRIÇÃO DOS ATRIBUTOS.

É possível observar os tipos dos atributos a partir figura 5, que apresenta uma parte da base de treinamento.

PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Transported
0001_01	Europa	False	B/P	TRAPPST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	Kiernan Clinchsky	False
0002_01	Earth	False	F/S	TRAPPST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	Juanna Vinos	True
0003_01	Europa	False	A/S	TRAPPST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	Adam Rosent	False
0003_02	Europa	False	A/S	TRAPPST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	Sidam Rosent	False
0004_01	Earth	False	F/S	TRAPPST-1e	16.0	False	303.0	70.0	151.0	965.0	2.0	Willy Swastardines	True
0005_01	Earth	False	F/P	PSO J1118.5-22	44.0	False	0.0	483.0	0.0	291.0	0.0	Sanderi Hovethrenes	True
0006_01	Earth	False	F/S	TRAPPST-1e	26.0	False	42.0	1539.0	3.0	0.0	0.0	Dilax Jacobstathry	True

Fig. 5. Algumas instâncias da base de treinamento.

#### C. Valores Ausentes

Para que os algoritmos de mineração de dados e de classificação atinjam o seus objetivos é necessário que as bases de dados treinamento e de teste não tenham valores nulos, portanto, é necessário identificá-los de antemão para depois realizar o tratamento dos dados. As tabelas II e III apresentam a quantidade de valores nulos de cada atributo para a base de treinamento e teste, respectivamente. Cabe ressaltar que a base de treinamento possui 2087 instâncias com

dados nulos, enquanto a base de teste possui 996 instâncias com valores ausentes, portanto em ambas as bases existem instâncias com mais de um atributo com valor ausente, além disso, ao observar as tabelas abaixo nota-se que quase todos os atributos assumem valores nulos em algumas instâncias.

Atributo	Quantidade de Valores Ausentes
PassengerId	0
HomePlanet	201
CryoSleep	217
Cabin	199
Destination	182
Age	179
VIP	203
RoomService	181
FoodCourt	183
ShoppingMall	208
Spa	183
VRDeck	188
Name	200
Transported	0

TABLE II  
VALORES AUSENTES NA BASE DE TREINAMENTO.

Atributo	Quantidade de Valores Ausentes
PassengerId	0
HomePlanet	87
CryoSleep	93
Cabin	100
Destination	92
Age	91
VIP	93
RoomService	82
FoodCourt	106
ShoppingMall	98
Spa	101
VRDeck	80
Name	94

TABLE III  
VALORES AUSENTES NA BASE DE TESTE.

#### IV. METODOLOGIA

Primeiramente foram feitas as seguintes transformações na base de treino e de teste:

- Separar Passenger ID em dois atributos: grupo (Group) e posição no grupo (PositionGroup).
- Separar Cabine em 3 atributos: Cabin\_Deck/ Cabin\_Num/ Cabin\_Side.
- Garantir que quem está hibernando (CryoSleep = *True*) não gaste nada.
- Garantir que passageiros com menos de 5 anos de idade não gastem nada.
- Criar novo atributo (Total\_Spent) sendo a soma dos gastos dos passageiros.
- Remover atributo Name.
- Realizar One-Hot-Encoding em PositionGroup, HomePlanet, CryoSleep, Cabin\_Deck, Cabin\_Side, Destination e VIP.
- Para o conjunto de treinamento, converter os valores booleanos de Transported para 0 (não transportado) e 1 (transportado).

Após essas transformações foi possível perceber que alguns valores ausentes foram preenchidos, no entanto, vários valores ainda estão nulos. Para resolver esse problema foi utilizado o KNNImputer com o valor de K igual a 5 ( $n\_neighbors=5$ ) em ambas as bases.

Seguidamente, as bases foram normalizadas utilizando a normalização MinMax. A qual obteve melhores resultados que qualquer combinação envolvendo a mesma e as normalizações StandardScaler e RobustScaler. Também foi implementado o algoritmo de análise de componentes PCA, e a técnica SMOTE para o balanceamento dos dados, mesmo sendo 50.4% para a classe *True* e 49.6% para a classe *False*, no entanto, só houve pioras na pontuação.

Para a classificação foram utilizadas um *ensemble* de 5 classificadores TabNet, todos com pré-treinamento não supervisionado, obtendo melhores resultados que a classificação sem o pré-treinamento. Cada modelo, tanto para o pré-treinamento quanto para o aprendizado supervisionado, foi ajustado de formas diferentes e treinado sobre porções distintas ou não da base de treinamento.

Para o pré-treinamento os parâmetros ajustados foram:

- *pretraining\_ratio*: porcentagem das características que são mascaradas durante o pré-treinamento. Ajustado em 0.8.
- *optimizer\_fn*: o otimizador. Ajustado para utilizar o Adam.
- *learning\_rate (lr)*: taxa de aprendizado inicial.
- *mask\_type*: tipo da máscara de seleção de características. Ajustada em *entmax*.
- *max\_epochs*: número máximo de épocas que o modelo passará no treinamento. Ajustado em 1000 épocas.
- *patience*: número de épocas consecutivas sem melhora antes da parada antecipada. Ajustado em 100.
- *batch\_size*: número de amostras por lote que serão carregadas. Ajustado em 256.
- *virtual\_batch\_size*: tamanho do lote virtual ( $virtual\_batch\_size \geq batch\_size$ ). Ajustado em 128.
- *num\_workers*: número de workers. Ajustado em 0.
- *drop\_last*: se deve descartar o último lote incompleto durante o treinamento. Ajustado com *False*.

Para o treinamento, além dos parâmetros supracitados, os que foram ajustados são:

- *step\_size*: quantidade de épocas que devem passar para aplicar gamma. Ajustado em 10.
- *gamma*: taxa de decaimento de aprendizado a cada *step\_size*. Ajustado em 0.9.
- *weights*: balanceamento das classes, 0 para não haver balanceamento, 1 para balanceamento automatizado, ou *dict* para pesos personalizados por classe. Ajustado em 1.

Os conjuntos de treino e validação foram avaliados com AUC (área sob a curva) e acurácia. A tabela IV abaixo apresenta os valores dos parâmetros variáveis para cada modelo em cada etapa.



Nº Modelo	Treino - Validação (%)	Modelo Não Supervisionado	Modelo Supervisionado
1	70% - 30%	lr=0.02	lr=0.02
2	70% - 30%	lr=0.01	lr=0.01
3	80% - 20%	lr=0.02	lr=0.02
4	80% - 20%	lr=0.01	lr=0.01
5	67% - 33%	lr=0.01	lr=0.01

TABLE IV  
PARÂMETROS DOS MODELOS.

Para avaliação do modelo *ensemble* foi selecionado por 20 vezes uma porcentagem aleatória da base de treino entre 10% e 90%, em seguida foi calculado da média e o desvio padrão dos resultados coletados (Figura 6).

	precision (s)	recall (s)	f1_score (s)
0	0.81252(0.00835)	0.877596(0.003617)	0.843785(0.005267)
1	0.870136(0.004656)	0.802067(0.00734)	0.834698(0.004919)
accuracy	0.83939(0.004783)	0.83939(0.004783)	0.83939(0.004783)

Fig. 6. Resultado do modelo *ensemble* na validação.

## V. RESULTADOS

Utilizando o modelo *ensemble*, realizando a predição no conjunto de teste e submetendo no Kaggle, foi possível atingir aproximadamente 80.41% de pontuação, sendo que, no momento, o top 1 possui 87.30% de pontuação e o top 5 conseguiu uma pontuação de 81.72% (Figura 7).


313	HALLIDAY GAUSS COSTA D OS SANTOS	0.80476	44	1s
 Your Best Entry! Your submission scored 0.80032, which is not an improvement of your previous score. Keep trying!				

Fig. 7. Resultado do modelo *ensemble* no Kaggle.

## VI. CONCLUSÃO

Conclui-se que esse trabalho foi de suma importância para verificar os resultados de uma DNN voltada para dados tabulares, além da diversão obtida ao participar da competição. Pretende-se melhorar a pontuação na competição, para isso, como trabalho futuro, deseja-se analisar quais as instâncias o modelo tem dificuldade em classificar para treinar um modelo sobre essas instâncias, realizando esse procedimento com e sem o one-hot-encoding. Também é pretendido utilizar o algoritmo XGBoost e comparar com a TabNet, para possivelmente realizar um *ensemble* sobre essas duas técnicas.

## REFERENCES

- [1] KAGGLE. Kaggle: Your Home for Data Science. Disponível em: <https://www.kaggle.com/>.
- [2] Solheim Bojer, Casper, and Jens Peder Meldgaard. "Kaggle forecasting competitions: An overlooked learning opportunity." arXiv e-prints (2020): arXiv-2009.
- [3] Ekin, S. İlhan Omurca, and Neytullah Acun. "A comparative study on machine learning techniques using Titanic dataset." 7th international conference on advanced technologies. 2018.
- [4] Batista, Gustavo Enrique de Almeida Prado. Pré-processamento de dados em aprendizado de máquina supervisionado. Diss. Universidade de São Paulo, 2003.
- [5] Silva-Ramírez, Esther-Lydia, et al. "Missing value imputation on missing completely at random data using multilayer perceptrons." Neural Networks 24.1 (2011): 121-129.
- [6] Armina, Roslan, et al. "A review on missing value estimation using imputation algorithm." Journal of Physics: Conference Series. Vol. 892. No. 1. IOP Publishing, 2017.
- [7] Arik, Sercan Ö., and Tomas Pfister. "Tabnet: Attentive interpretable tabular learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 8. 2021.
- [8] Kothari Yash. "Predicting the Survivors of the Titanic -Kaggle, Machine Learning From Disaster". Academia.edu, 2018.
- [9] Alessandro, Lombardi, Zacchei Filippo Supervisors Polvani Niccolo, and Krapp Lucien Fabrice. "Mechanism of Action (MoA) Prediction-Kaggle Competition."