Universidade do Minho

Dados e Aprendizagem Automática

Checkpoint

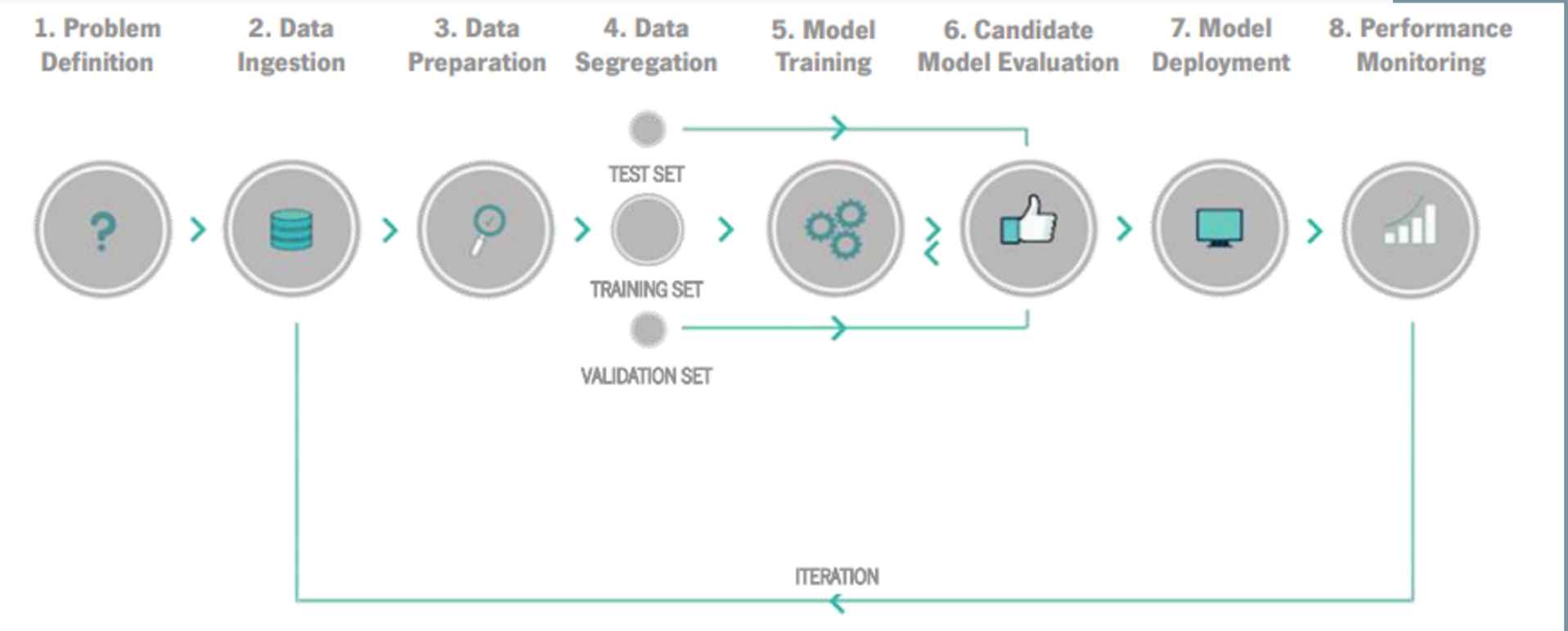
Eduardo Cunha

Jorge Rodrigues

João Magalhães

Rodrigo Gomes

Metodologia



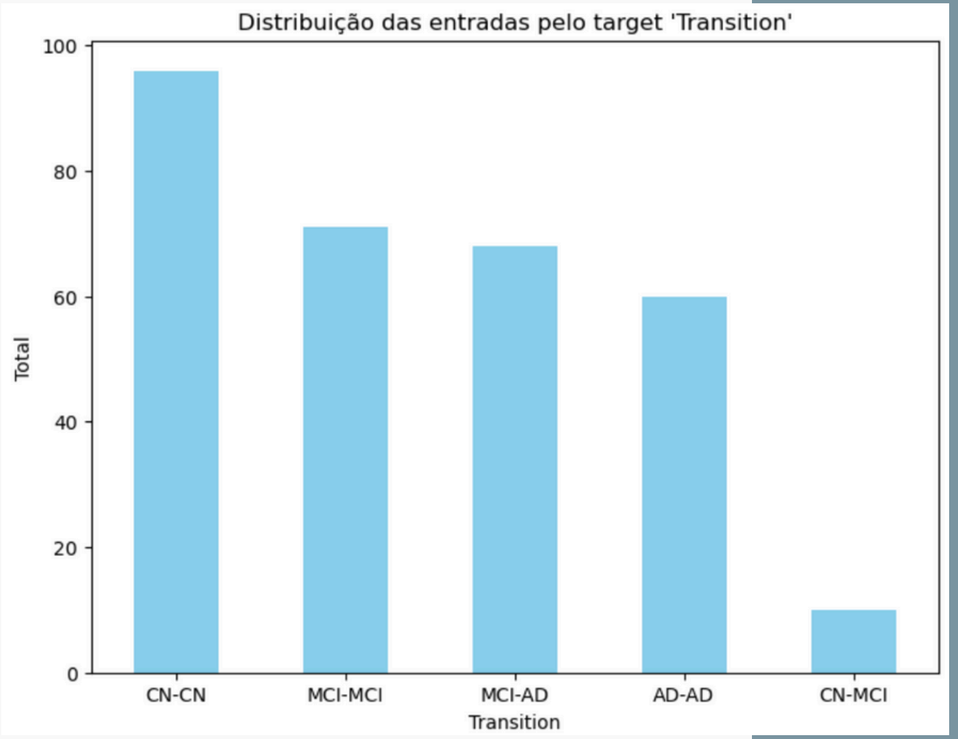
Data Exploration

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 305 entries, 0 to 304
Columns: 2181 entries, ID to Transition
dtypes: float64(2014), int64(147), object(20)
memory usage: 5.1+ MB
```

Não há entradas duplicadas

Não há valores em falta

A grande maioria das colunas são numéricas



Data Preperations

Prep 1

- Drop de colunas identificadoras
- Drop a colunas não numéricas
- Normalização MinMax

Prep 2

- Drop de colunas identificadoras
- Drop a colunas não numéricas
- Drop a colunas constantes
- Outlier Removal Z-score
- Normalização Standard

Prep 3

- Drop de colunas identificadoras
- Drop a colunas não numéricas
- Drop a colunas constantes
- Normalização MinMax

Prep 4

- Drop de colunas identificadoras
- Drop a colunas não numéricas
- Normalização MinMax
- Feature Selection

Outras Data Preparations Testadas

Ao longo do trabalho, experimentámos outras abordagens de data preparation que não foram destacadas, pois considerámos que originaram resultados inferiores, nomeadamente:

PCA (Principal Component Analysis)

- Técnica utilizada para a redução da dimensionalidade dos dados, com o objetivo de transformar um conjunto de variáveis correlacionadas num conjunto mais reduzido de variáveis não correlacionadas.

SMOTE (Synthetic Minority Over-sampling Technique)

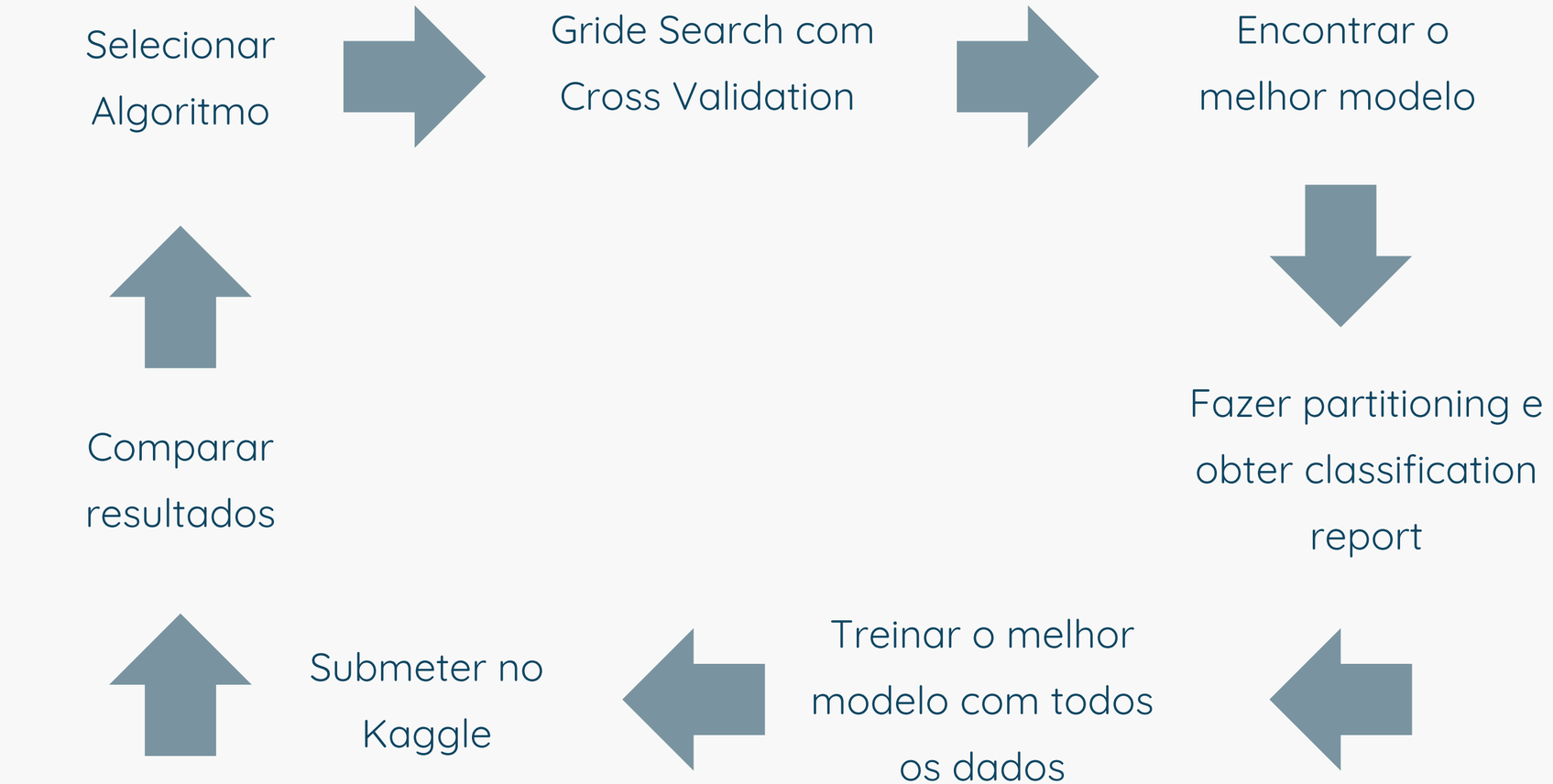
- Técnica utilizada para resolver problemas de desbalanceamento em conjuntos de dados, em que uma ou mais classes apresentam significativamente menos exemplos em comparação com as restantes.

Remover colunas com elevada correlação

- Variáveis altamente correlacionadas podem introduzir redundância e dificultar tanto a análise quanto o desempenho do modelo.

Models

Ciclo de Treino dos Modelos



Resultados Antigos

ID / Nome	score kaggle	f1	Descrição	Preparação
stacking v1	0.26487	0.42	modelos usados: decisionTreeClassifier, SVC, randomforest	prep.py
stacking v2	/	0.39	modelos usados: decisionTreeClassifier, SVC, randomforest	Prep: ID1
max_voting v1	/	0.40	modelos usados: decisionTreeClassifier, SVC, randomforest || voting = "hard" || weights = 2,1,2	Prep: ID1
RandomForestGrid1.0	/	0.39	Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000], "max_depth": [5, 10, 20, None],"criterion":["gini","entropy"] Melhores Parametros Estimados: 'criterion': 'gini', 'max_depth': 5, 'n_estimators': 300	Prep: ID1
BaggingGrid1.0	0.31494	0.39	Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000] Melhores Parametros Estimados: 'n_estimators': 1000	Prep: ID1
GradientBoostingGrid1.0	0.35987	0.45	Utilizei uma Grid para estimar a melhor combinação de parametros:"n_estimators": [500, 600, 800], "learning_rate": [0.1, 0.3]" Melhores Parametros Estimados: 'learning_rate': 0.3, 'n_estimators': 500	Prep: ID1
XGB1.0	/	0.38	Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [500,800,1000], "learning_rate": [0.1,0.01], "max_depth": [6,10,20], Melhores Parametros Estimados: 'learning_rate': 0.1, 'max_depth': 20, 'n_estimators': 500	Prep: ID1
SVM1.0	/	0.4	Grid: param_grid = 'C': [0.1, 1, 10, 100], ['linear', 'rbf', 'poly'], gamma': ['scale', 'auto'] Escolhidos: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 500}	Prep: ID1
Voting1.0	/	0.43	Modelos utilizados: GB peso 3, SVM peso 2, RF peso 1	Prep: ID1
RandomForestGrid2.0	/	0.4	Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000], "max_depth": [5, 10, 20, None],"criterion":["gini","entropy"] Melhores Parametros Estimados: {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 100}	Prep: ID2
BaggingGrid2.0	/	0.44	Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000] Melhores Parametros Estimados: 'n_estimators': 500	Prep: ID2
GradientBoostingGrid2.0	/	0.51	Utilizei uma Grid para estimar a melhor combinação de parametros:"n_estimators": [500, 600, 800], "learning_rate": [0.1, 0.3]" Melhores Parametros Estimados: {'learning_rate': 0.3, 'n_estimators': 500}	Prep: ID2
XGB2.0	/	0.43	Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [500,800,1000], "learning_rate": [0.1,0.01], "max_depth": [6,10,20], Melhores Parametros Estimados:{'learning_rate': 0.3, 'max_depth': 6, 'n_estimators': 500}	Prep: ID2
			Grid: param_grid = 'C': [0.1, 1, 10, 100], ['linear', 'rbf', 'poly'], gamma': ['scale', 'auto']	

Resultados Antigos

			Grid: param_grid = 'C': [0.1, 1, 10, 100], ['linear', 'rbf', 'poly'], gamma': ['scale', 'auto']	
SVM2.0	/	0.43	Escolhidos: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 500}	Prep: ID2
Voting2.0	/	0.49	Modelos utilizados: GB peso 3, SVM peso 2, RF peso 1	Prep: ID2
			Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000]	
BaggingGrid3.0	/	0.32	Melhores Parametros Estimados: 'n_estimators': 500	Prep: ID3
			Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000], "max_depth": [5, 10, 20, None],"criterion":["gini","entropy"]	
RandomForestGrid3.0	/	0.37	Melhores Parametros Estimados: {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 500}	Prep: ID3
			Utilizei uma Grid para estimar a melhor combinação de parametros:"n_estimators": [500, 600, 800], "learning_rate": [0.1, 0.3]"	
GradientBoostingGrid3.0	/	0.31	Melhores Parametros Estimados: learning_rate': 0.1, 'n_estimators': 500'	Prep: ID3
			Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [500,800,1000], "learning_rate": [0.1,0.01], "max_depth": [6,8,10,20],	
XGB3.0	/	0.32	Melhores Parametros Estimados: {'learning_rate': 0.1, 'max_depth': 8, 'n_estimators': 500}	Prep: ID3
			Grid: param_grid = 'C': [0.1, 1, 10, 100], ['linear', 'rbf', 'poly'], gamma': ['scale', 'auto']	
SVM3.0	/	0.33	Escolhidos:{'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}	Prep: ID3
Stacking3.0	/	0.36	estimators = [("gb", gb_grid.best_estimator_), ("svm", svm_grid.best_estimator_), ("rf", rf_grid.best_estimator_)]	Prep: ID3
max_voting v3	/	0.36	modelos usados: decisionTreeClassifier, SVC, randomforest || voting = "hard" || weights = 2,1,2	Prep: ID3
			Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000]	
BaggingGrid4.0	/	0.37	Melhores Parametros Estimados: 'n_estimators': 500	Prep: ID3
			Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [100, 500, 800, 1000], "max_depth": [5, 10, 20, None],"criterion":["gini","entropy"]	
RandomForestGrid4.0	/	0.31	Melhores Parametros Estimados: {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 100}	Prep: ID4
			Utilizei uma Grid para estimar a melhor combinação de parametros:"n_estimators": [500, 600, 800], "learning_rate": [0.1, 0.3]"	
GradientBoostingGrid4.0	/	0.28	Melhores Parametros Estimados: learning_rate': 0.01, 'n_estimators': 500'	Prep: ID4
			Utilizei uma Grid para estimar a melhor combinação de parametros: "n_estimators": [500,800,1000], "learning_rate": [0.1,0.01], "max_depth": [6,8,10,20],	
XGB4.0	/	0.31	Melhores Parametros Estimados: {'learning_rate': 0.3, 'max_depth': 5, 'n_estimators': 500}	Prep: ID4
			Grid: param_grid = 'C': [0.1, 1, 10, 100], ['linear', 'rbf', 'poly'], gamma': ['scale', 'auto']	
SVM4.0	/	0.31	Escolhidos: {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}	Prep: ID4
Stacking4.0	/	0.28	estimators = [("gb", gb_grid.best_estimator_), ("svm", svm_grid.best_estimator_), ("rf", rf_grid.best_estimator_)]	Prep: ID4
max_voting v4	/	0.28	modelos usados: decisionTreeClassifier, SVC, randomforest || voting = "hard" || weights = 2,1,2	Prep: ID4

Neural Network

Neural Network 1

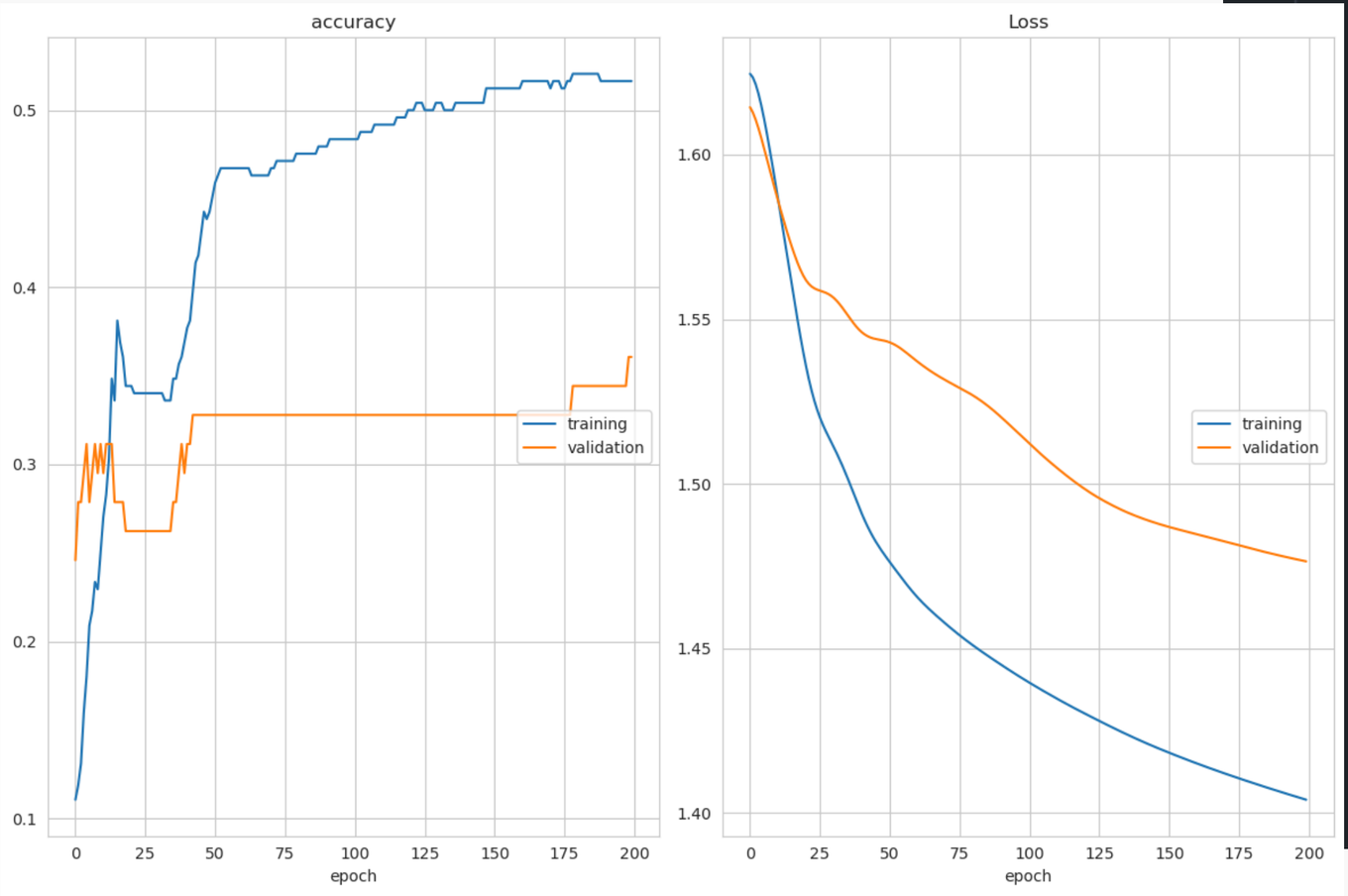
25/60 certas

```python
class Model(nn.Module):
    def __init__(self, in_features=2161, h1=2000, h2=1500, h3=1000, h4=500, out_f
        super().__init__()
        self.fc1 = nn.Linear(in_features, h1)
        self.fc2 = nn.Linear(h1, h2)
        self.fc3 = nn.Linear(h2, h3)
        self.fc4 = nn.Linear(h3, h4)
        self.out = nn.Linear(h4, out_features)

    def foward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = F.relu(self.fc4(x))
        x = self.out(x)

        return x
```

Neural Network 2




```
from torch.nn import Module, Linear, ReLU, Softmax, Sigmoid
from torch.nn.init import xavier_uniform_, kaiming_uniform_

class MLP_1(Module):
    def __init__(self, n_inputs):
        super(MLP_1, self).__init__()
        self.hidden1 = Linear(n_inputs, 1000)
        kaiming_uniform_(self.hidden1.weight, nonlinearity='relu')
        self.act1 = ReLU()
        self.hidden2 = Linear(1000, 500)
        kaiming_uniform_(self.hidden2.weight, nonlinearity='relu')
        self.act2 = ReLU()
        self.hidden3 = Linear(500, 5)
        xavier_uniform_(self.hidden3.weight)
        self.act3 = Softmax(dim=1)

    def forward(self, X):
        X = self.hidden1(X)
        X = self.act1(X)
        X = self.hidden2(X)
        X = self.act2(X)
        X = self.hidden3(X)
        X = self.act3(X)
        return X
```

```
Accuracy: 0.361

success:22 failure:39
```

Relativamente ao Trabalho

Consideramos que o objetivo do projeto está a ser cumprido e acreditamos que estamos num bom caminho. Identificámos um desafio significativo relacionado com o tamanho do dataset. Com apenas 100 entradas disponíveis, torna-se extremamente difícil desenvolver um modelo que consiga alcançar bons resultados, uma vez que conjuntos de dados tão pequenos limitam a capacidade do modelo de generalizar e aprender padrões relevantes.

Esta limitação é especialmente evidente no caso das redes neuronais. Estes modelos exigem grandes quantidades de dados para treinar eficazmente os seus parâmetros e capturar padrões complexos.

Trabalho Futuro

Ciclo de Treino dos Modelos-Novo

Resultados Novos

Random Forest							
		Parametros					
Preparação	Teste	n_estimators	max_depth	criterion	max_features	F1-macro	Kaggle
1	Cross validation k = 5	100	5	entropy	log2	0.3529707241982752	0.39298
3	Cross validation k = 5	100	20	entropy	log2	0.3369315523878811	0.31972

Gradient Boosting								
		Parametros						
Preparação	Teste	n_estimators	max_depth	learning_rate	max_features	loss	F1-macro	Kaggle
1	Cross validation k = 5	100	5	0.1	sqrt	log_loss	0.36091994615642997	0.27301
3	Cross validation k = 5	50	5	0.3	none	log_loss	0.3453177006157634	0.40015

Bagging				
		Parametros		
Preparação	Teste	n_estimators	F1-macro	Kaggle
1	Cross validation k = 5	100	0.33572302338674587	/
3	Cross validation k = 5	100	0.32542731139639997	/

XGBoosting							
		Parametros					
Preparação	Teste	n_estimators	max_depth	learning_rate	eval_metric	F1-macro	Kaggle
1	Cross valic	50	5	0.1	mlogloss	0.3484980	/
3	Cross valic	50	5	0.1	mlogloss	0.3484980	/

SVM						
		Parametros				
Preparação	Teste	C	kernel	gamma	F1-macro	Kaggle
1	Cross valic	1	linear	scale	0.3146322329963119	/
3	Cross valic	1	rbf	scale	0.2974620454453589	/

Universidade do Minho

Dados e Aprendizagem Automática

Checkpoint

Eduardo Cunha

Jorge Rodrigues

João Magalhães

Rodrigo Gomes