Mestrado em Engenharia Informática

Dados e Aprendizagem Automática
Grupo 19

Relatório Trabalho Prático

Aluno/a:	*Número:*
Ana Luísa Carneiro	PG46983
Ana Rita Peixoto	PG46988
Henrique Guimarães Ribeiro	A89582
Luís Miguel Pinto	PG47428

Janeiro 2022

Conteúdo

1 Introdução 2
2 Exploração dos *datasets* 3
2.1 Previsão do fluxo de tráfego rodoviário 3
Remoção de Colunas 3
Transformações de valores categóricos 4
Criação de *Features* - Data da observação 4
Tratamento de *outliers* 4
Normalização os dados 6
2.2 Previsão da probabilidade de ocorrer um AVC 6
Remoção de Colunas 6
Transformação do tipo de valores 7
Transformação de valores nulos 7
Agrupar Categorias - BMI 7
Tratamento de *outliers* 7
Normalização os dados 8
3 Criação dos modelos 8
3.1 Previsão do fluxo de tráfego rodoviário 8
DecisionTreeClassifier 9
RandomForestClassifier 9
AdaBoostClassifier 10
BaggingClassifier 11
LogisticRegression 11
GaussianNB 12
VotingClassifier 12
3.2 Previsão da probabilidade de ocorrer um AVC 13
DecisionTreeClassifier 13
RandomForestClassifier 13
4 Obtenção e análise crítica de resultados 14
4.1 Previsão do fluxo de tráfego rodoviário 14
4.2 Previsão da probabilidade de ocorrer um AVC 17
5 Sugestões e Recomendações 18
5.1 Previsão do fluxo de tráfego rodoviário 18
5.2 Previsão da probabilidade de ocorrer um AVC 18
6 Conclusão 20

1 Introdução

Neste trabalho realizado no âmbito da UC de Dados e Aprendizagem Automática foi proposta a exploração, modelação e análise de dois *datasets*: um focado na previsão do tráfego rodoviário numa dada área da cidade do Porto e outro à escolha de cada grupo de trabalho. Para o segundo caso, o *dataset* escolhido pelo nosso grupo está relacionado com a previsão de ocorrer AVC em indivíduos, tendo em conta um dado número de atributos.

Para o primeiro *dataset* é nos proposto prever o nível de tráfego,*AVERAGE_SPEED_DIFF* (*target* do problema), nas suas várias categorias, *None*, *Low*, *Medium*, *High* e *Very_High*, para cada observação tendo em conta diversos atributos. Para o segundo dataset é nos proposto prever a probabilidade (*Yes* ou *No*) de um individuo ter um AVC, *Stroke* (*traget* do problema) segundo vários atributos ligados à saúde e estilo de vida de um paciente.

O presente relatório apresenta uma análise detalhada e justificada de todo o procedimento efetuado em cada um dos *datasets* até à aplicação dos modelos de *Machine Learning*. É possível observar quais os métodos de visualização e exploração de dados, assim como os modelos de aprendizagem aplicados e os respetivos resultados. Além disso, o presente documento também irá cobrir os testes submetidos na plataforma *kaggle* que permitiu ter uma melhor noção da precisão dos modelos e implicações de certas mudanças no tratamento de dados.

Como forma de obtermos, eficientemente, uma análise completa e robusta dos *datasets* decidiu-se aplicar ao desenvolvimento do projeto uma metodologia **CRISP-DM**. Esta abordagem de mineração de dados é composta por 5 fases: **entender o negócio** (avaliação do contexto do problema), **entender os dados** (analisamos todos os atributos e seu significado e contexto), **preparação dos dados** (tratamento dos dados), **modelação** (desenvolvimento de modelos de *machine Learning*), **avaliação** (análise dos resultados obtidos) e por fim, **implementação** (aplicação do tratamento e modelação realizados). Esta última fase não foi realizada pois não se encontra dentro do contexto do projeto.

2 Exploração dos *datasets*

O primeiro passo da realização deste trabalho prático consistiu em explorar os *datasets* de forma a prepara-los para a aplicação de modelos de *Machine Learning*. Além disso, a exploração dos dados também auxilia na escolha do modelo de extração de conhecimento mais adequado. A aplicação de técnicas de preparação dos dados é essencial dado que os *datasets* do mundo real contém muitas inconsistências, podem ser incompletos e conter lixo.

A primeira ação tomada para cada um dos *datasets* na exploração dos dados foi a **observação do *dataset*** e a **análise de cada atributo** que o constitui, observando a contagem dos registos de cada um e de alguns dados estatísticos como: a média, desvio padrão e percentil dos dados numéricos.

Nesta secção é possível observar qual o tratamento de dados aplicado a cada *dataset* e os métodos de visualização e exploração de dados que conduziram a uma dada decisão.

2.1 Previsão do fluxo de tráfego rodoviário

Remoção de Colunas

A determinação da importância de cada atributo no *dataset* consistiu em **analisar a natureza dos valores de cada atributo**: no caso de atributos categóricos, avaliar quais os possíveis valores de este pode tomar; no caso de atributos numéricos, entender como estes variam ou até mesmo se são nulos. Com esta análise pudemos concluir que a coluna *city_name*, que identificava o nome da cidade em questão, era constituída pelo mesmo valor para todos os registos e, portanto, não iria contribuir para um aumento da precisão dos resultados. Deste modo, procedemos à remoção da coluna *city_name* do *dataset*. Analogamente, o valor do atributo *AVERAGE_PRECIPITATION* demonstrou-se constante ao longo de todos os registos e igual a zero. Deste modo, esta coluna *AVERAGE_PRECIPITATION* também foi removida, dado que não iria acrescentar informação relevante e útil ao modelo de aprendizagem.

Além disso, a contagem dos registos associados a cada atributo permitiu-nos observar que apenas existiam 8% de registos para o atributo *AVERAGE_RAIN*, sendo que 92% estavam em falta. Dado que a percentagem de registos preenchidos era muito reduzida, a previsão dos valores desconhecidos não seria muito fiável e iria trazer ruído para o modelo. Por esta razão, e de modo a reduzir dados que iriam perturbar o modelo, decidimos remover a coluna *AVERAGE_RAIN*.

Uma outra coluna que possuía valores em falta era *AVERAGE_CLOUDINESS* que também foi removida. A remoção desta coluna necessitou de maior análise dado que a percentagem de valores não preenchidos era cerca de 39%. Devido a ser um valor moderado, foram efetuados vários testes para perceber se este atributo estaria a ter um impacto considerável no modelo, desde a interpolação dos valores em falta até à consideração de que os valores em falta equivaliam à inexistência de nuvéns. Após terem sido realizadas as análises e vários testes, concluímos que a sua influência no modelo não era de facto significativa e que estava a trazer mais ruído à previsão, logo

decidimos remover esta coluna.

Por fim, procedemos também à remoção da coluna *AVERAGE_HUMIDITY* após observar uma forte correlação com o atributo *AVERAGE_TEMPERATURE*. Esta alta correlação significa que o modelo está a aprender a mesma coisa com os dois atributos, tornando um deles redundante.

Transformações de valores categóricos

Um outro passo realizado no tratamento dos dados do *dataset* foi a transformação dos valores do tipo categórico em valores numéricos. Esta transição deveu-se essencialmente à incompatibilidade que alguns modelos de aprendizagem automática tem com valores do tipo categórico. Deste modo, os atributos que sofreram esta transformação foram: *AVERAGE_SPEED_DIFF* e *LUMINOSITY*. Apesar de ter existido uma transição para valores numéricos, esta foi meramente representativa e não retirou o devido significado aos valores de cada atributo.

Criação de *Features* - Data da observação

No que diz respeito à criação de novas *features* este processo apenas foi aplicado à coluna *record_date* com o objetivo de conseguir obter mais informações dos dados através de uma melhor organização dos mesmos. Neste sentido, procedemos a sua divisão em mês, dia da semana e hora do dia. Notar que apesar de o ano estar presente na data original, este foi menosprezado uma vez que não consideramos ter grande influência no modelo. Além disso, o dia da semana foi representado de forma numérica em que cada valor equivale a cada dia de domingo a sábado, sequencialmente.

Relativamente a data da observação foram ainda realizados testes acerca do contributo de características ligadas ao atributo *Day*, como por exemplo: altura do dia (manhã, horário de trabalho, fim-de-tarde e noite) ou ser feriado - os quais concluímos não ter grande relevância para a precisão do modelo. Também consideramos o agrupamento de certas características da data como por exemplo : a coluna dos meses passaria a estar identificada consoante a estação do ano e a coluna relativa ao *Day* passaria a identificar se o o dia em questão era fim-de-semana. Após análise, verificou-se que o primeiro método não relevou ser muito impactante para o modelo, contudo o segundo método teve bastante influência na precisão do modelo.

Tratamento de *outliers*

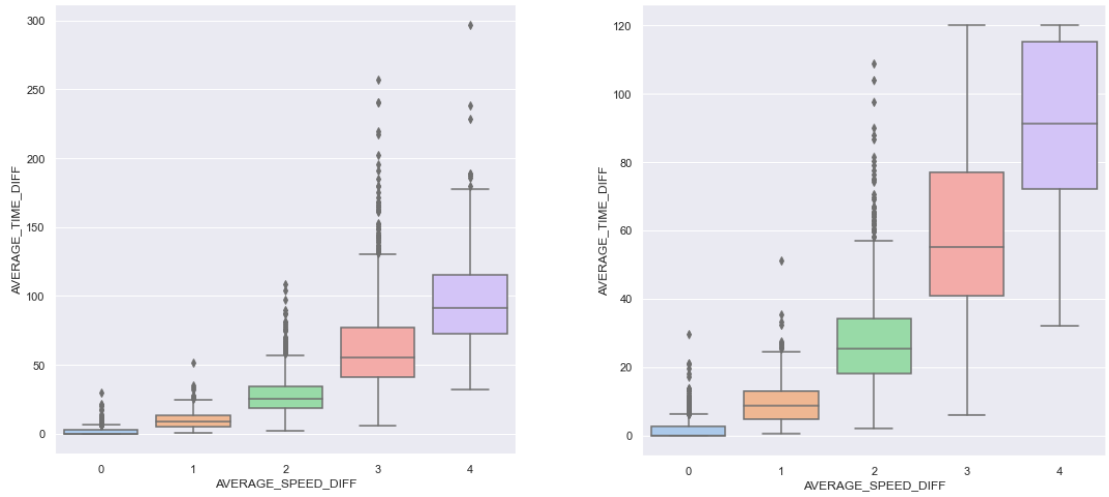
Relativamente aos *outliers* existentes no *dataset* foram efetuados diversos tratamentos. Após a aplicação de métodos de visualização e exploração de dados, podemos observar que existiam diversos *outliers* em vários atributos principalmente no atributo *AVERAGE_TIME_DIFF*, que é o atributo com maior com maior correlação com o *target* do nosso problema.

Numa primeira fase retiramos os *outliers* que estavam acima de um determinado valor. Contudo, este tratamento não teve os resultados desejados e por isso em vez de

os retirar, decidiu-se transformá-los na média do atributo para que estes se encontrem dentro do intervalo de valores do *AVERAGE_TIME_DIFF*.

Numa segunda fase, o tratamento dos dados consistiu em efetuar 2 previsões: uma previsão para um conjunto de dados sem *outliers* e outra previsão para o conjunto de *outliers* deste atributo. Assim, começou-se por dividir o *dataset* de treino e teste em dois conjuntos de dados através de uma função *'is_outlier'* (imagem 1.2 a)) que determina se um valor é um *outlier*. Deste modo, dividiu-se os dados entre aqueles valores que são e os que não são *outilers*. De seguida, efetuou-se a previsão dos resultados através da previsão separada de cada um dos conjuntos de dados. Ao utilizar este método os valores de previsão do modelo melhoraram de forma significativa.

Numa terceira e última fase decidiu-se aproximar os *outliers* ao valor do quartil mais próximo e para isso utilizou-se uma função *'fence'* (imagem 1.2 b)) que determina a *outter e upper fence* de cada atributo. Esta barreira permite identificar quais os *outilers* desse atributo, por exemplo quando um valor está acima da *upper fence* então esse valor é um *outlier*. De seguida os valores que estão para lá das barreiras vão ser aproximados ao valor do quartil mais próximo para assim ficarem dentro do intervalo de valores do atributo. Este método para além de ter sido aplicado ao atributo *AVERAGE_TIME_DIFF* também acabou por ser aplicado aos restantes atributos contínuos. Este método está representado na figura 1.1.



(a) Antes do tratamento

(b) Depois do tratamento

Figura 1.1: Tratamento de *outilers dataset* da competição

```python
def is_outlier(s):
    lower_limit = s.mean() - (s.std() * 1)
    upper_limit = s.mean() + (s.std() * 1)
    return ~s.between(lower_limit, upper_limit)
```

(a) Função *is_outlier*

```python
#Outer fences of the variable CRIM
from scipy.stats.mstats import winsorize
def fences(traffic, variable_name):
    q1 = traffic[variable_name].quantile(0.25)
    q3 = traffic[variable_name].quantile(0.75)
    iqr = q3-q1
    outer_fence = 3*iqr
    outer_fence_le = q1-outer_fence
    outer_fence_ue = q3+outer_fence
    return outer_fence_le, outer_fence_ue
```

(b) Função *fences*

Figura 1.2: Funções para o tratamento de *outilers*

Normalização os dados

A normalização dos dados do *dataset* foi também objeto de análise e teve como propósito retirar peso a valores mais altos devido às unidades em que estes se encontram. Certos modelos são muito influenciados por altos valores no *dataset* e por isso, convém que haja normalização para que esses modelos não dependam tanto das unidades de medida. A normalização consistem em transformar os valores dos atributos entre 0 e 1, sendo 0 o valor mais baixo do tributo e 1 o mais alto. Desta forma, damos a mesma importância a todos os atributos do *dataset*.

Para se normalizar os valores do *dataset* foi utilizado o *min_max_scaler.fit_transform* que normalizada valores contínuos e o *get_dummies* que transforma os valores categóricos em colunas com valores entre 0 e 1.

2.2 Previsão da probabilidade de ocorrer um AVC

Remoção de Colunas

O primeiro passo consistiu em **analisar a natureza dos valores de cada atributo**: no caso de atributos categóricos, avaliar quais os possíveis valores de este pode tomar; no caso de atributos numéricos, entender como estes variam ou até mesmo se são nulos. Desta forma, reparamos que na coluna *bmi* havia alguns valores a NULL (201), contudo a quantidade destes valores era reduzida quando comprada com todos os valores na coluna *bmi*, não havendo por isso necessidade de remover esta coluna.

Além disso, a partir do contexto do problema e dos atributos reparamos que a coluna *id* que representa o identificador único do paciente é redundante para o problema uma vez que não influencia em nada para a probabilidade de ocorrer um AVC. Desta forma, decidiu-se retirar esta coluna.

Transformação do tipo de valores

Durante a análise e exploração dos dados reparou-se que os dados na coluna *age* eram do tipo *float* o que não fazia sentido no contexto do problema, uma vez que a idade não é um valor contínuo mas sim um valor inteiro. Desta forma, transformou-se o tipo de dados *float* em dados *int*.

Transformação de valores nulos

Tal como já foi mencionado na secção de remoção de colunas, na coluna *bmi* existem 201 valores a *null* que foram preenchidos utilizando a média dos 4909 valores. Desta forma, os valores que foram preenchidos não ficam desajustados, não criam *outliers* nem grandes variações nos valores da coluna.

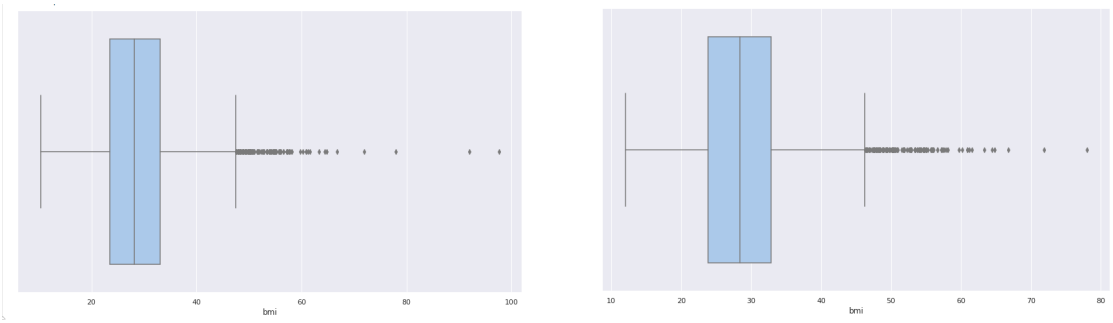
Agrupar Categorias - BMI

Para o atributo *bmi* decidiu-se agrupar os valores em categorias, pois assim torna-se melhor a classificação de cada paciente de acordo com a sua massa global, retirando influência dos valores do *bmi* e dando mais peso à classificação de cada paciente. Desta forma foram utilizada as seguintes categorias:

- ≤ 18.5 : *underweight* (falta de massa corporal)
- 18.5 - 24.9 : normal (massa corporal normal)
- 25 - 29.9: *overweight* (massa corporal acima da média)
- 30 - 34.9: *obese* (obeso)
- ≥ 35 : *extremely obese* (extremamente obeso)

Tratamento de *outliers*

A partir da exploração e visualização dos dados reparamos que existem *outliers* muito pouco realistas para o contexto do atributo na coluna *bmi* (por exemplo valores de massa corporal próximos dos 100). Desta forma, através de uma pesquisa, averiguou-se que valores abaixo de 12 e acima de 78 não são dados realistas para representar a *bmi* de um paciente. Assim, achou-se conveniente transformar esse intervalo de valores na média da coluna *bmi*, mitigando a influência dos *outliers* no nosso problema. Na figura abaixo consegue-se observar as diferenças da escala entre os dados com tratamento e sem tratamento de dados.



(a) Antes do tratamento (b) Depois do tratamento

Figura 1.3: Tratamento de *outilers dataset* escolhido

Normalização os dados

A normalização dos dados do *dataset* foi também objeto de análise e teve como propósito retirar peso a valores mais altos devido às unidades em que estes se encontram. Certos modelos são muito influenciados por altos valores no *dataset* e, por isso, convém que haja normalização para que esses modelos não dependam tanto das unidades de medida. A normalização consistem em transformar os valores dos atributos entre 0 e 1, sendo 0 o valor mais baixo do atributo e 1 o mais alto. Desta forma, damos a mesma importância a todos os atributos do *dataset*.

Para se normalizar os valores do *dataset* foi utilizado o *min_max_scaler.fit_transform* que normalizada valores contínuos ou inteiros (*avg_glucose_level*, *bmi* e *age*) e o *get_dummies* que transforma os valores categóricos em colunas com valores entre 0 e 1 (restantes colunas).

3 Criação dos modelos

O segundo passo na realização do projeto foi o desenvolvimento de modelos de *Machine Learning* como forma de dar resposta ao problema proposto em cada *dataset*. Todos os modelos desenvolvidos utilizaram o tratamento de dados realizado na primeira etapa do trabalho.

Nesta fase apresentamos a descrição e características dos vários modelos desenvolvidos em cada *dataset* assim como o *tunning* de parâmetros realizado em cada modelo.

3.1 Previsão do fluxo de tráfego rodoviário

Como forma de testar os modelos e tratamentos dos dados sem submissão no *Kaggle* utilizou-se a função *cross_val_score* que através dos atributos para modelação e do *target* determina a precisão no *dataset* de treino. Nas imagens abaixo podemos ver a implementação dos modelos sendo que a variávell $\boldsymbol{x}$ representa os dados da modelação, a variável $\boldsymbol{y}$ representa o *target* e a variável $\boldsymbol{test}$ representa o conjunto de dados de teste cujo as previsões serão submetidas na competição Kaggle.

Dentro dos sistemas de aprendizagem, os algoritmos adequados à resolução deste problema encontram-se na categoria de **Aprendizagem com Supervisão**, mais especificamente na categoria de **Classificação**, pois trata-se de um problema de previsão de valores categóricos.

Na presente secção estão expostos todos os algoritmos testados para este *dataset* e as respetivas conclusões retiradas a partir de cada um.

DecisionTreeClassifier

O primeiro algoritmo utilizado para a previsão do modelo foi o *DecisionTreeClassifier* que se trata de um algoritmo baseado em árvores de decisão constituídas por nodos e folhas, e funciona através da recursividade através dos nodos. Na figura 1.4 está representado a implementação da *DecisionTreeClassifier* no contexto do problema.

```
print("**DecisionTreeClassifier**")
clf_Tree = DecisionTreeClassifier(random_state=2021)
clf_Tree.fit(x,y)

print("Test Data...")
predictions_Tree = clf_Tree.predict(test)
print(predictions_Tree)

#k cross val
print("Training Data...")
scores = cross_val_score(clf_Tree,x,y,cv = 10)
print("Cross Validation Accuracy: %0.4f (+/- %0.4f)" % (scores.mean(), scores.std() * 2))
```

Figura 1.4: *DecisionTreeClassifier* - Dataset Competição

RandomForestClassifier

Como tentativa de otimizar os resultados obtidos no algoritmo anterior, utilizamos o *RandomForestClassifier* que também é baseado em árvores de decisão. A diferença para o algoritmo *DecisionTreeClassifier* é que o *Random Forest* não se apoia numa única decisão. Este considera decisões aleatórias baseadas em múltiplas árvores de decisão e baseia a sua decisão na maioria. Através da inserção de um fator aleatório, conseguimos alcançar maior diversidade e maior precisão. Na figura 1.5 conseguimos ver a implementação da RandomForestClassifier.

```
print("**RandomForestClassifier**")
clf_Forest = RandomForestClassifier(n_estimators=100, random_state=2021)
clf_Forest.fit(x,y)

print("Test Data...")
predictions_Forest = clf_Forest.predict(test)
print(predictions_Forest)

print("Training Data...")
scores = cross_val_score(clf_Forest,x,y,cv = 10)
print("Cross Validation Accuracy: %0.4f (+/- %0.4f)" % (scores.mean(), scores.std() * 2))
```

Figura 1.5: *RandomForestClassifier* - Dataset Competição

Como este algoritmo foi aquele com que obtivemos melhores valores de precisão e previsão dos dados, então decidiu-se utilizar o ***Grid Search*** como forma de obter os melhores hiperparâmetros do modelo que façam com que este atinja valores mais altos de precisão. Os hiperparâmetros que foram alterados de forma a obter melhores resultados foi o *n_estimator* que indica o número de estimadores (árvores) no modelo e o *criterion* que representa a qualidade de um *split*. Apesar de termos realizado o *Grid Search* com outros hiperparâmetros, estes foram aqueles que mais contribuíram para o aumento da precisão com este modelo.

Para além da implementação da *Grid Search* também foi implementado a ***Feature Selection*** que faz com que o modelo selecione os atributos mais relevantes para fazer a modelação, rejeitando aqueles atributos que tem pouca influência no problema.

Finalmente, também foi aplicado a este modelo o método *Nested Cross Validation* que tem como objetivo avaliar e comparar o *tunning* de modelos de *machine learning*. Para isso cria-se dois *KFold*, um que vai servir para fazer o *tunning* dos hiperparâmetros e outro que vai implementar o *RandomForestClassifer* com os melhores hiperparâmetros e consequentemente obter a média da precisão desse modelo. Este segundo *KFold* é usado para obter a precisão com o *dataset* de treino e para saber se o modelo está a ter *overfitting* ou não. Para determinamos a previsão com o *dataset* de teste basta aplicar este conjunto de dados ao modelo com os melhores hiperparâmetros obtido com o primeiro *KFold*. Na imagem 1.6 está representado o método *Nested Cross Validation*.

```
cv_inner = KFold(n_splits=3, shuffle=True, random_state=1)
# define the model
model = RandomForestClassifier(random_state=2021)
# define search space
space = dict()
space['n_estimators'] = [10, 100, 500]
space['max_features'] = [2, 4, 6]
space['criterion'] = ['gini', 'entropy']
# define search
search = GridSearchCV(model, space, scoring='accuracy', n_jobs=1, cv=cv_inner, refit=True)

result = search.fit(x,y)
best_model = result.best_estimator_
predictions= best_model.predict(test)
# configure the cross-validation procedure
#predictions_final  = pd.DataFrame(predictions, columns = ["Speed_Diff"])
#print(predictions_final)
cv_outer = KFold(n_splits=10, shuffle=True, random_state=1)
# execute the nested cross-validation
scores = cross_val_score(search, x, y, scoring='accuracy', cv=cv_outer, n_jobs=-1)
# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

Figura 1.6: *Nested Cross Validation* - Dataset Competição

AdaBoostClassifier

Um outro algoritmo que também foi alvo de testes, foi o *AdaBoostClassifier*. Este algoritmo é um método de *ensemble* que opera ao colocar maior peso em atributos difíceis de classificar e menos peso em atributos que já são fáceis de prever. Os resultados obtidos através da utilização deste algoritmo mantiveram-se na mesma escala que os resultados obtidos com o *Random Forest*. Na figura 1.7 podemos ver a implementação do modelo *AdaBoostClassifier*.

```
print("**AdaBoostClassifier**")
base = RandomForestClassifier(n_estimators=500, random_state=2021, criterion='entropy')
clf_Booster = AdaBoostClassifier(n_estimators=100, random_state=2021, base_estimator=base)
clf_Booster.fit(x,y)

print("Test Data...")
predictions_Booster = clf_Booster.predict(test)
print(predictions_Booster)

#k cross val
print("Training Data...")
scores = cross_val_score(clf_Booster,x,y,cv = 10)
print("Cross Validation Accuracy: %0.4f (+/- %0.4f)" % (scores.mean(), scores.std() * 2))
```

Figura 1.7: *AdaBoostClassifier* - Dataset Competição

BaggingClassifier

O BaggingClassifier foi também um algoritmo diferente usado. Este algoritmo também é um método de *ensemble*, que gera vários conjuntos de dados através de *bootstrapping*. Estes conjuntos servem depois para treinar várias instâncias do mesmo modelo de aprendizagem que depois são agregadas por maioria ou média.

No nosso caso, testamos com o algoritmo *K-nearest neighbours* e com o *DecisionTreeClassifier*, apresentando os dois níveis precisões menores em comparação ao *RandomForestClassifier*. Na figura 1.8 conseguimos ver a implementação deste modelo.

```
print("**BaggingClassifier**")
clf_BG = BaggingClassifier(KNeighborsClassifier(), random_state=2021, max_samples=0.5, max_features=0.5)
clf_BG.fit(x,y)

print("Test Data...")
predictions_BG = clf_BG.predict(test)
print(predictions_BG)

#k cross val
print("Training Data...")
scores = cross_val_score(clf_BG,x,y,cv = 10)
print("Cross Validation Accuracy: %0.4f (+/- %0.4f)" % (scores.mean(), scores.std() * 2))
```

Figura 1.8: *BaggingClassifier* - Dataset Competição

LogisticRegression

A *LogisticRegression* é um método de aprendizagem baseado no algoritmo de regressão logística, que usa uma função logística para produzir um modelo que permita a previsão de valores de uma variável binário ou categórica a partir de uma série de variáveis. Na figura 1.9 podemos vera implementação deste método.

Este modelo apresentou uma precisão consideravelmente mais baixa do que qualquer outro dos modelos referidos.

GaussianNB

As *Naive Bayes* são um grupo de algoritmos de aprendizagem baseados no teorema de *Bayes*. Um dos algoritmos que experimentamos foi o *Gaussian Naive Bayes*, que aplica o teorema de *Bayes* que assume a independência entre parâmetros.

Em termos de resultados, este algoritmo produziu resultados com um nível de precisão abaixo do *Ada Boost* e do *Random Forest* e equivalente ao algoritmo Decision Trees. Na figura 1.9 podemos vera implementação deste método.

VotingClassifier

O VotingClassifier é um método de ***ensemble*** que usa um conjunto de modelos na classificação. O classificador por votos pode ser estabelecido por voto maioritário ou probabilidades médias previstas na classificação. No nosso caso, usamos voto maioritário.

Este modelo proporcionou um nível de precisão abaixo do *Ada Boosting* e do *Random Forest* mas acima dos restantes modelo usados. Na figura 1.9 podemos ver a implementação deste método, assim como todos os modelos que foram utilizados por ele.

```
print("**Voting Classifier**")
# clf1 = DecisionTreeClassifier(random_state=2021)
# clf2 = RandomForestClassifier(n_estimators=100, random_state=2021, criterion='entropy')
# clf3 = BaggingClassifier(KNeighborsClassifier(), random_state=2021, max_samples=0.5, max_features=0.5)
# clf4 = AdaBoostClassifier(n_estimators=100, random_state=2021, base_estimator=base)
clf5 = LogisticRegression(random_state=1)
clf6 = GaussianNB()

eclf = VotingClassifier(
    estimators=[('DecisionTree', clf_Tree), ('RandomForest', clf_Forest2), ('bag', clf_BG),
                ('boost', clf_Booster), ('lr', clf5), ('gnb', clf6)],
    voting='hard')

#evaluate the test dataset
#test_p = eclf.predict(X_test)
#test_acc = accuracy_score(Y_test,test_p)

for clf, label in zip( [clf_Tree, clf_Forest2, clf_BG, clf_Booster, clf5, clf6, eclf], ['Decision Tree',
    'Random Forest', 'Bagging', 'Boosting', 'Logistic Regression', 'naive Bayes', 'Voting Ensemble']):
    scores = cross_val_score(clf, x, y, scoring='accuracy', cv=5)
    print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

Figura 1.9: *VotingClassifier* - Dataset Competição

3.2 Previsão da probabilidade de ocorrer um AVC

Como forma de testar os modelos e tratamentos utilizou-se a função *cross_val_score* que através dos atributos para modelação e do *target* determina a precisão no *dataset*. Nas imagens abaixo podemos ver a implementação dos modelos sendo que a variável $\boldsymbol{x}$ representa os dados da modelação, a variável $\boldsymbol{y}$ representa o *target*.

Dentro dos sistemas de aprendizagem, os algoritmos adequados à resolução deste problema encontram-se na categoria de **Aprendizagem com Supervisão**, mais especificamente na categoria de **Classificação**, pois trata-se de um problema de previsão da probabilidade (sim ou não) de ocorrer de AVC.

DecisionTreeClassifier

O primeiro algoritmo utilizado para a previsão do modelo foi o *DecisionTreeClassifier* que se trata de um algoritmo baseado em árvores de decisão constituídas por nodos e folhas, e funciona através da recursividade através dos nodos. Na figura 1.10 podemos ver a implementação deste modelo no contexto do problema.

```
modelDecisionTree = DecisionTreeClassifier()
scores = cross_val_score(modelDecisionTree, x, y, cv=10)
print("Result for Decision Tree: %0.4f accuracy with standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

Figura 1.10: *DecisionTreeClassifier* - Dataset Grupo

RandomForestClassifier

Como tentativa de otimizar os resultados obtidos no algoritmo anterior, utilizamos o *RandomForestClassifier* que também é baseado em árvores de decisão. A diferença para o algoritmo *DecisionTreeClassifier* é que o *Random Forest* não se apoia numa única decisão. Este considera decisões aleatórias baseadas em múltiplas árvores de decisão e baseia a sua decisão na maioria. Através da inserção de um fator aleatório, conseguimos alcançar maior diversidade e maior precisão. Na figura 1.10 podemos ver a implementação do modelo.

```
modelRandomForest = RandomForestClassifier(n_estimators = 500)
scores = cross_val_score(modelRandomForest, x, y, cv=10)
print("Result for Random Forest: %0.4f accuracy with standard deviation of %0.2f" % (scores.mean(), scores.std()))
```

Figura 1.11: *RandomForestClassifier* - Dataset Grupo

4 Obtenção e análise crítica de resultados

A terceira etapa do trabalho consiste na análise dos resultados obtidos com os modelos de *machine learning*. Desta forma, apresentamos um sumário dos resultados obtidos com cada *dataset* e a respetiva análise crítica.

4.1 Previsão do fluxo de tráfego rodoviário

Para a obtenção e análise crítica de resultados no *dataset* da competição, iremos basear a nossa abordagem na explicação de cada submissão na plataforma *Kaggle*. Assim, para cada submissão constará a razão pela qual a mesma foi efetuada, quais os resultados esperados, os resultados obtidos e os motivos das eventuais diferenças entre os resultados.

Desta forma conseguiremos obter, por um lado, uma melhor noção do contributo de cada modelo, bem como o impacto dos refinamentos e *tunnings* efetudos sobre os mesmos. Não obstante, a interpretação dos resultados permitirá ainda avaliar a forma como as modificações efetuadas no dataset influenciam a solução final.

Visto que nesta secção iremos abordar o processo de obtenção de resultados, fará sentido apresentar os resultados de *accuracy*, presentes na *pontuação pública* - que representa 30 % do *dataset* de teste, aos quais o grupo tinha acesso durante a realização das submissões, uma vez que essa era métrica guia e motivadora que impulsionava e orientava o desenvolvimento do trabalho. Assim sendo, deixaremos para a próxima secção - "Sugestões e Recomendações " - a análise da *pontuação privada*.

Na imagem seguinte podemos ver tanto a coluna da pontuação pública (*public score*) como da pontuação privada (*private score*), sendo que os valores da pontuação privada surgiram após a criação e teste dos modelos pelo que não serão tidos em conta na explicação do raciocínio por detrás de cada submissão.

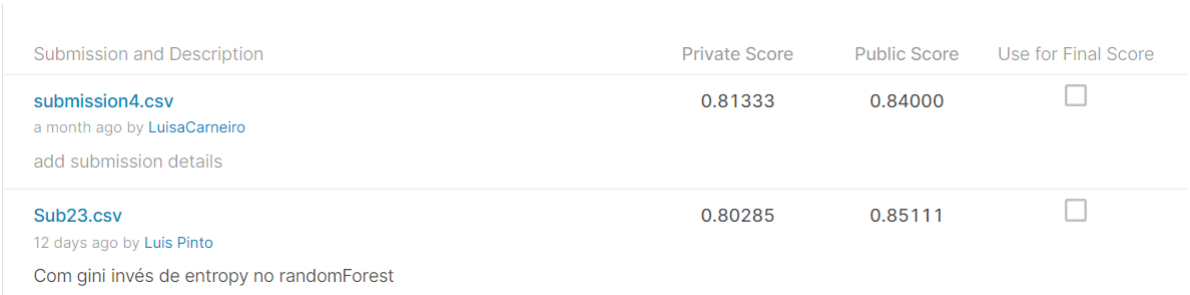


Figura 1.12: Exemplo da página submissões na competição do Kaggle

No decorrer da competição foram submetidos 28 ficheiros de previsão, alguns com mudanças mais significativas, outros mais refinados e minuciosos com pequenas alterações. De seguida procederemos a análise pontual de cada caso :

- Submissão 1 : O primeiro modelo a ser testado utilizou **Árvores de Classificação**. O principal objetivo desta submissão e da submissão 2 foi avaliar o contributo do atributo *AVERAGE_CLOUDINESS*. Deste modo, nesta submissão o atributo foi removido e a sua precisão foi de **74.888 %** .

- Submissão 2 : Na segunda submissão mantivemos o modelo anterior (*DecisionTreeClassifier*) e o atributo *AVERAGE_CLOUDINESS*, a sua precisão foi de **73.777 %** . Pelo que concluímos que para a estimação do modelo seria melhor, por enquanto, não considerar o atributo relativo a nebulosidade média.

- Submissão 3 : Lapso da equipa, o ficheiro submetido foi o da 1 Submissão. Nas próximas submissões foram efetuados os testes pretendidos.

- Submissão 4 : Implementação do modelo utilizando **RandomForestClassifier**. O principal objetivo nas submissões 4 e 5 seriam o de testar as diferença inerentes a mudança de modelo. Ou seja, a submissão 4 será semelhante a submissão mas com outro modelo. Comparativamente as versões com *DecisionTreeClassifier* a precisão obtida foi maior, tal como era previsto, sendo o seu valor de **84.666 %** .

- Submissão 5 : Tal como já foi mencionado, usamos novamente o modelo **RandomForestClassifier**, esta submissão pode ser equiparada a submissão 2 no sentido em que mantivemos o atributo ligado a nebulosidade. O seu valor de *accuracy* foi de **84.000 %** , que comprova novamente a hipótese de ser melhor excluir a coluna do atributo *AVERAGE_CLOUDINESS*.

- Submissão 6 : Tendo chegado a conclusão de que o **RandomForestClassifier** seria um modelo mais confiável, chegamos agora ao ponto de teste de **hiperparâmetros** e ***tunning*** do modelo. Assim sendo, através de mecanismos de **gridSearch** chegamos à conclusão de que devíamos aumentar o número de estimadores de 100 para 500 e de mudar o critério de divisão de *giny* (por omissão) para *entropy* de forma a otimizar ainda mais o modelo. Como já teria sido comprovado que o critério de nebulosidade média não favorecia os resultados obtidos este não foi considerado. O verdadeiro objetivo desta submissão seria agora avaliar o impacto do atributo *AVERAGE_HUMIDITY*, o qual foi removido para efeitos de teste. Tal como era expectável, tanto pela otimização dos parâmetros do modelo como pela suspeitável pouca influência do atributo relacionado com a humidade média, a precisão subiu para os **84.666 %** .

- Submissão 7 : Nesta submissão, dando continuidade a submissão anterior, manteve-se o modelo de **RandomForest**, testando-se agora o efeito da **normalização** dos valores e a **remoção de outliers** através de métodos previamente mencionados. A precisão obtida foi de **73.111 %**, contrariamente a previsão do grupo uma vez que diminuiu.

- Submissão 8 : Tendo explorado bastante os modelos anteriormente referidos, o grupo procurou testar um novo modelo recorrendo ao **AdaBoostClassifier**, **não**

removendo os outliers mas efetuando novamente a **normalização**. O valor obtido foi de **78.000 %** , o que levou a querer que a escolha da remoção dos outliers não foi a melhor maneira de lidar com eles.

- Submissão 9 : No sentido de tentar tratar de uma melhor forma os outliers nesta submissão tudo se mantêm igual comparativamente a submissão anterior, apenas com a diferença de que os **outliers foram substituídos** invés de removidos. O valor de precisão não aumento, nem diminuiu, mantendo-se nos **78.000 %** .

- Submissão 10, Submissão 11, Submissão 12, Submissão 13 : Testes realizados sobre a **seleção / divisão dos outliers**, bem como qual o melhor mecanismo de substituição do seu valor, tal como esta descrito na secção 2.1 (tratamento dos *outliers*). As suas precisões foram, respetivamente : **85.111 %** , **84.222 %** , **83.777 %** e **84.000 %**. Isto levou a adotar as métricas da submissão 10.

- Submissão 14 : Nesta submissão o objetivo principal foi avaliar a **influência de ser ou não fim de semana**. A sua precisão foi de **85.555 %** , sendo uma melhoria significativa e passando esta métrica a ser utilizada em submissões futuras.

- Submissão 16 : Nesta submissão considerou-se novamente o atributo ligado à **humidade** para testar se o seu impacto era positivo. A precisão atingida foi de **85.111 %**, pelo que não se revelou melhor ao obtido anteriormente.

- Submissão 17 : Nesta submissão mais uma vez tentou-se identificar quais seriam os outliers agora recorrendo a um **intervalo confiança de 95 %** para a sua escolha. A precisão obtida com este teste foi de **85.333 %** .

- Submissão 20 : Nesta submissão usou-se a técnica de **one hot encoding** para atributos como a chuva, nebulosidade e iluminação através da função **get_dummies** tal como já foi mencionado anteriormente. A precisão obtida foi de **84.000 %**.

- Submissão 21 , Submissão 22 : Foram efetuados testes removendo as colunas relativas à humidade e à iluminação, de modo a analisar a sua influência no modelo. A precisão obtida em ambas foi de **85.111 %**.

- Submissão 23 : Igual à submissão 14 mas com **critério *gini*** invés de *entropy* no ***RandomForest***. A precisão obtida foi de **85.111 %**.

- Submissão 19, Submissão 24 : Nestas submissões foi utilizada a técnica de **Feature Selection** juntamente com o modelo de **RandomForest**. A precisão obtida foi de **85.333 %**.

- Submissão 15, Submissão 18, Submissão 25 : Estas submissões serviram todas um propósito idêntico, o de implementar o ***Nested Crossed Validation*** tal como foi mencionado anteriormente na secção de criação dos modelos. Na submissão 25 este método foi devidamente implementado pelo que obtivemos o melhor de precisão, **86.000 %** .

- Submissão 26 : Tendo agora por base o **Nested Cross Validation** as submissões seguintes (26, 27 e 28) surgem todas com o propósito de melhorar ainda mais a precisão da submissão através do tratamento dos dados. Nesta submissão

identificaram-se os outliers e **substituiu-se o seu valor pelo percentil 99 %** . A precisão obtida foi de **85.777 %** .

- Submissão 27 : Nesta submissão efetuou-se a mesma divisão dos *outliers* tal como está na submissão 10, contudo o tratamento dos *outliers* nos restantes atributos foi efetuado consoante a **aproximação ao percentil mais próximo**. A precisão obtida foi de **85.777 %** .

- Submissão 28 : Nesta submissão efetuou-se o **agrupamento pelas horas** do dia segundo intervalos horários bem definidos. A *accuracy* tem o valor de **86.000 %** .

Notar que apesar de terem sido desenvolvidos e testados outros modelos de aprendizagem (*BaggingClassifier*, *LogisticRegression*, *VotingClassifier*, entre outros), apenas os que apresentavam maior potencial (i.e, *RandomForestClassifier* e *AdaBoostClassifier*) é que foram submetidos na plataforma *Kaggle*.

4.2 Previsão da probabilidade de ocorrer um AVC

Para a análise crítica dos resultados obtidos no *dataset* escolhido pela equipa, iremos basear a nossa abordagem nos resultados obtidos com os vários tratamentos e modelos que foram implementados no *dataset*. Para isso, dividiu-se o *dataset* em treino e teste para que primeiro conjunto de dados seja utilizado na modelação e o segundo seja utilizado na obtenção de previsões e resultados.

Numa primeira fase implementou-se o modelo *DecisionTreeClassifier* aplicado ao tratamento de dados tal como foi mencionado na secção 3.2. No entanto nesta primeira fase não se agrupou o *bmi* em categorias nem se tratou os *outliers*. Com esta modelação e tratamento atingiu-se a precisão de 91,25% de precisão. Ao mesmo tratamento também foi aplicado o modelo *RandomForestClassifier* onde se obteve uma precisão 95,03%. Isto deve se ao facto de que o *DecisionTreeClassifier* utiliza apenas uma árvore para a classificação dos valores e o *RandomForestClassifier* utiliza múltiplas para fazer o mesmo processo. Assim concluímos que este segundo modelo é melhor para ser aplicado neste contexto.

Numa segunda fase implementou-se os mesmos modelos só que aplicados a outra tipo de tratamento de dados. Nesta exploração aplicamos para além do tratamento realizado na fase anterior, o agrupamento do *bmi* tal como está na secção 3.2. Nesta fase obtivemos uma precisão de 90,53% para o modelo *DecisionTreeClassifier* e de 94,76% para o modelo de *RandomForestClassifier*.

Numa terceira e última fase implementou-se os mesmos modelos só que aplicados a todo o tratamento de dados tal como está na secção 3.2. Nesta fase obtivemos uma precisão de 90,71% para o modelo *DecisionTreeClassifier* e de 94,76% para o modelo de *RandomForestClassifier*.

5 Sugestões e Recomendações

A última etapa consiste na análise dos resultados obtidos no contexto do problema assim como a resposta ao problema proposto por cada *dataset*. Desta forma, apresentamos qual foi, a nosso ver, o tratamento de dados e respetivo modelo que melhor responderam a cada um dos problemas.

5.1 Previsão do fluxo de tráfego rodoviário

Neste problema propunha-se a previsão do fluxo de tráfego rodoviário numa dada área na cidade do Porto, desta forma considera-se um bom modelo aquele que prevê o tráfego com maior precisão.

Durante a realização do trabalho o raciocínio e as decisões que foram implementadas basearam-se no resultado obtido com 30% dos dados de teste na plataforma *Kaggle*. Contudo, após o encerramento da competição determinou-se que a submissão com melhor previsão para 100% dos dados de teste foi a submissão 4. Nesta submissão utilizou-se o modelo *RandomForestClassifier* e um tratamento de dados onde se removeu a coluna *AVERAGE_CLOUDINESS*. Ainda nesta submissão, manteve-se a coluna da humidade e os *outliers* não foram tratados nem foi implementada a normalização.

De forma a responder a este problema sugere-se que seja implementado o modelo *RandomForestClassifier* uma vez que foi o modelo com que obtivemos maior precisão e recomenda-se um tratamento de dados onde se remova a coluna *AVERAGE_CLOUDINESS* tal como foi explicado na secção 2.1 no tópico de remoção de colunas. Por comparação com os resultados obtidos na Submissão 6, também recomenda-se que se mantenha a coluna *AVERAGE_HUMIDITY* que apesar de ter alta correlação com a coluna *AVERAGE_LUMINOSITY*, ainda tem influência na previsão de tráfego e por isso não é conveniente removê-la do *dataset*. Ao contrário daquilo que se previa também os *outliers* do atributo *AVERAGE_TIME_DIFF* são importantes para classificar as várias observações. Além disso, como o modelo *RandomForestClassifier* não é influenciado por intervalo de valores elevados, ao contrário de modelos como o *Support Vector Machine*, a normalização não tem qualquer influência no modelo.

5.2 Previsão da probabilidade de ocorrer um AVC

Neste problema propunha-se a previsão da probabilidade (sim ou não) de ocorrer de um AVC num dado paciente, desta forma considera-se um bom modelo aquele que previne de forma correta se um paciente tem ao não probabilidade de ter um AVC.

Na secção 5.2 apresentamos as diferentes modelações e tratamento assim como o resultado obtido com cada, desta forma conseguimos verificar que o teste com maior precisão e consequente melhor previsão foi com o modelo *RandomForestClassifier* sem tratamento de *outilers* e sem agrupamento do *bmi*.

De forma a responder a este problema sugere-se que seja implementado o modelo *RandomForestClassifier* uma vez que foi o modelo com que obtivemos maior precisão

(95,03%) e recomenda-se um tratamento de dados onde não se agrupe as várias categorias de *bmi*, pois assim retiramos a influencia dos valores no modelo. Ao contrário daquilo que se previa também os *outliers* do atributo *bmi* são importantes para classificar as várias observações.

6 Conclusão

Dado por concluído o projeto, fará sentido apresentar uma visão crítica, refletida e ponderada do trabalho realizado.

A realização deste trabalho permitiu-nos consolidar a matéria leccionada na cadeira de DAA, especialmente no que diz respeito ao tratamento de dados e desenvolvimento de modelos de *Machine Learning*.

No espetro positivo, destacamos o uso de diversos modelos de *machine learning* e técnicas para tratamento de dados desde *one-hot-encoding*, *feature selection* e *feature creation* até a normalização e tratamento de *outliers*. Em adição, consideramos que o uso de tecnologias novas, não lecionadas no contexto curricular, demonstra o nosso interesse e dedicação pelo trabalho realizado, em particular (referimo-nos) a implementação de modelos que fazem parte do *ensemble*, como o *RadomForestClassifier*, *AdaBoost*, entre outros.

Relativamente a pontos a melhorar, concluímos que apesar da vasta gama de modelos de *machine learning* utilizados gostaríamos de ter explorado outras vertentes no futuro como por exemplo testar diferentes modelos com o uso de *stacking* ou criar uma nova implementação de aprendizagem através do uso de redes neuronais, desta forma saciando a nossa curiosidade pela temática e aprofundando o nosso conhecimento.

Consideramos que o presente documento se encontra explicativo, estando de acordo com as etapas estabelecidas na metodologia adotada, CRISP-DM. Além disso, o conjunto de modelos implementados é completo e a exploração realizada consegue responder de forma eficaz e correta aos problemas evidenciados nos *datasets*.

Em suma, consideramos que o balanço do trabalho é positivo, as dificuldades sentidas foram superadas e os requisitos propostos foram cumpridos.