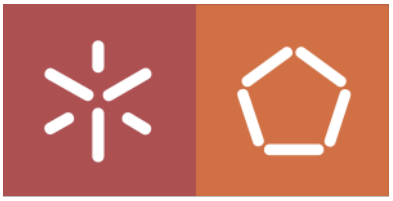

Universidade do Minho
Escola de Engenharia

UMinho

Mestrado Engenharia Informática
Dados e Aprendizagem Automática (2022/23)

Grupo: 40

pg50229 António Luís de Macedo Fernandes
pg50483 João Paulo Sousa Mendes
pg50499 João Silva Torres
pg50518 José Diogo Martins Vieira



Braga, January 15, 2023

Contents

1 Introdução

No âmbito do desenvolvimento do projeto da unidade curricular Dados e Aprendizagem Automática foi-nos proposto desenvolver um projeto de Machine Learning utilizando, entre outros, os modelos de aprendizagem abordados ao longo do semestre.

Dito isto, tivemos de trabalhar com dois datasets: o de grupo e o da competição. No primeiro, o grupo fez uma análise e selecionou, a partir de fontes externas (Kaggle), o *music_genre.csv*. Quanto ao segundo, a equipa docente disponibilizou o dataset "previsão de Acidentes Rodoviários" a todos os grupos de trabalho com o objetivo de formar, tal como diz o nome, uma competição.

De seguida, procedemos à exploração, análise e preparação de ambos os modelos, com o objetivo de extrair conhecimento relevante no contexto do problema. Realizamos também conceção e otimização de múltiplos modelos de Machine Learning.

Ao longo deste relatório iremos explicar, com rigor, todas as técnicas e resultados obtidos.

Na última secção faremos uma análise crítica sobre todo o trabalho desenvolvido pelo grupo.

2 Tarefas

2.1 DATASET GRUPO

Numa primeira fase, através da fonte externa Kaggle, selecionamos o nosso dataset que é relativo a previsões de géneros musicais. (https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre).

2.1.1 Quais os domínios a tratar, quais os objetivos e como se propõem a atingi-los

O dominio deste dataset (**prediction-of-music-genre**) é musical e é, inicialmente constituído por 18 colunas e cerca de 50000 entradas. Cada entrada respresenta uma música diferente. Nas colunas são apresentados valores para diversos paramentros relacionados com música, por exemplo, duração, musicalidade, energia, entre outros.

O objetivo deste modelo é prever, através destes atributos, qual o estilo de música (rock, techno, etc) de cada entrada. Trata-se então de um modelo de classificação.

Selecionamos este dataset por ser referente a uma área de interesse do grupo e por acharmos interessante a previsão de um estilo musical dadas as caracteristicas da própria.

2.1.2 Qual a metodologia seguida e como foi aplicada

A metodologia que foi aplicada para o tratamento deste dataset foi CRISP-DM. Esta metodologia divide-se em várias fases, sendo elas:

- **Business Understanding:** Inicialmente começamos então por definir e entender, o problema e contexto do nosso dataset assim como o seu objetivo. Neste caso trata-se da previsão de estilos musicais utilizando diferentes métricas relativas ás diferentes músicas
- **Data Understanding:** Nesta fase tratamos de recolher as várias entradas de dados do dataset referente (prediction-of-music-genre.csv) e de os explorar. Isto é, verificar a qualidade dos mesmos e que tipos de tratamentos que terão de ser feitos de forma a os tornar melhores
- **Data Preparation:** Nesta fase passamos a preparação de dados e o pré processamento dos mesmos antes da modelção. Sendo este tratamento realizado através da eliminação de valores duplicados, tratamento de missing values, outliers, ente outros. O intuito desta fase é, facilitar a previsão dos vários modelos de aprendizagem sobre os dados.
- **Modeling** Nesta fase pusemos em prática vários modelos de Machine Learning sobre os nossos dados. Foi respetido este processo até achar o melhor modelo de treino
- **Evaluation:** Nesta fase final fez-se uma avalição da performance dos vários modelos e seleciona-se o melhor para o objetivo pretendido.

2.1.3 Descrição e exploração detalhada do dataset e do tratamento efetuado

Nesta secção iremos abordar como foi feita a exploração do nosso dataset e o tratamento feito sobre os dados do mesmo.

Verificámos, primeiro a existência de missing values.

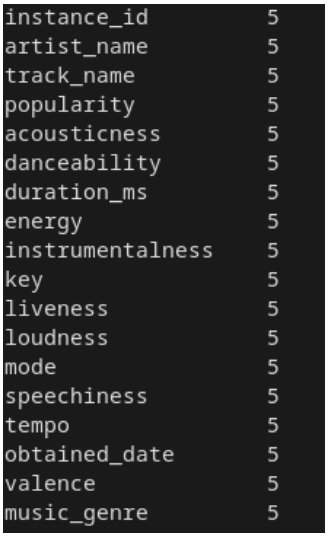

Figure 1: Quantidade de missing values para cada genero musical

Assim sendo, decidimos eliminar todos os *missing values* existentes no dataset.

```
df = df.dropna(axis=0)
```

Verificámos também a inexistência de registos duplicados através da função:

```
df.duplicated().sum())
```

Figure 2: Quantidade de duplicados para cada género musical

De modo a não desperdiçar por completo a informação contida na coluna "track_length", criamos uma nova feature chamada track name length que terá o tamanho do nome da música, pois poderá ajuda a prever o seu estilo musical (*FeatureEngennering*).

```
df['track_name_length'] = df['track_name'].apply(lambda x: len(x))
```

De seguida eliminamos as seguintes *features*: *instance_ id*, *obtained_ date*,*artist_ name* e *track_ name*, porque não possuem informação que contribuam para alcançar o objetivo. Tanto o id como a obtained_date são irrelevantes, pois o id é relacionado com o número de identificação da música de uma plataforma, a obtained_date é relativa à data que o autor do dataset obteve a informação.

```
df.drop(['instance_id','obtained_date','artist_name','track_name'],axis=1)
```

Posteriormente, decidimos agrupar o estilo de musica "Rap" e "Hip-Hop" por serem estilos musicais muito idênticos e muito difícil de os distinguir dadas as features do nosso dataset.

```
df['music_genre'] = df['music_genre'].apply(lambda x : 'Rap' if x == 'Hip–Hop' else x)
```

Assim como, os estilos musicais, Jazz e Blues.

```
df['music_genre'] = df['music_genre'].apply(lambda x : 'Jazz/Blues'
if x == 'Jazz' or x== 'Blues' else x)
```

Ademais, decidimos eliminar o genero musical *"Alternative"* porque é um estilo ambíguo e, por isso, difícil de descobrir o mesmo.

```
df = df[df.music_genre != 'Alternative']
```

Adiante, na tratamento dos *missingvalues* presentes no feature tempo e duração, aplicamos a média de forma a preencher os mesmos, passando o tipo de string para float.

```
# handle missing values on tempo
df['tempo'] = df['tempo'].apply(lambda x : 0 if x == '?' else float(x))
mean = df['tempo'].mean()
df['tempo'] = df['tempo'].apply(lambda x : mean if x == 0 else float(x))
```

```
mean = df['duration_ms'].mean()
df['duration_ms'] = df['duration_ms'].apply(lambda x : mean if x == -1 else x)
```

Para além disso, eliminamos a *feature "energy"* uma vez que esta é bastante correlacionada com outras features, como veremos a seguir na matriz de correlação.

```
df = df.drop(['energy'], axis=1)
```

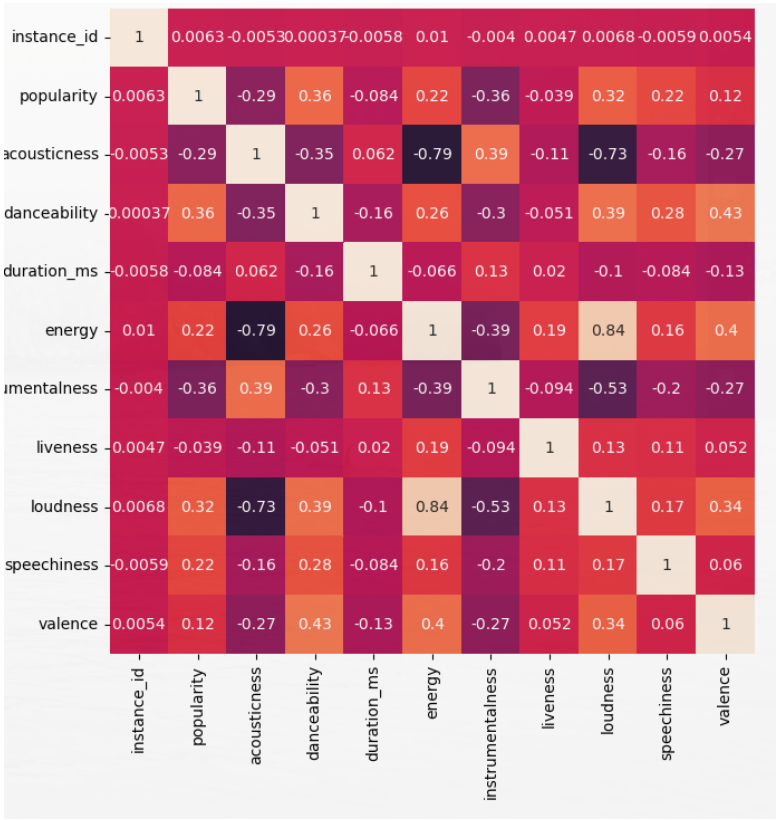

Figure 3: Matriz de correlação antes do tratamento de dados

Também foi feito o *label encoding* das features *key*, *mode*, que são valores categóricos, de forma a passa-las a valores numéricos.

```
df['key'] = lb_make.fit_transform(df['key'])

#Label encoding mode
df['mode'] = df['mode'].apply(lambda x : 1 if x == 'Major' else 0)
```

Por fim criamos a matriz de correlação dos *features* do nosso *dataset*.

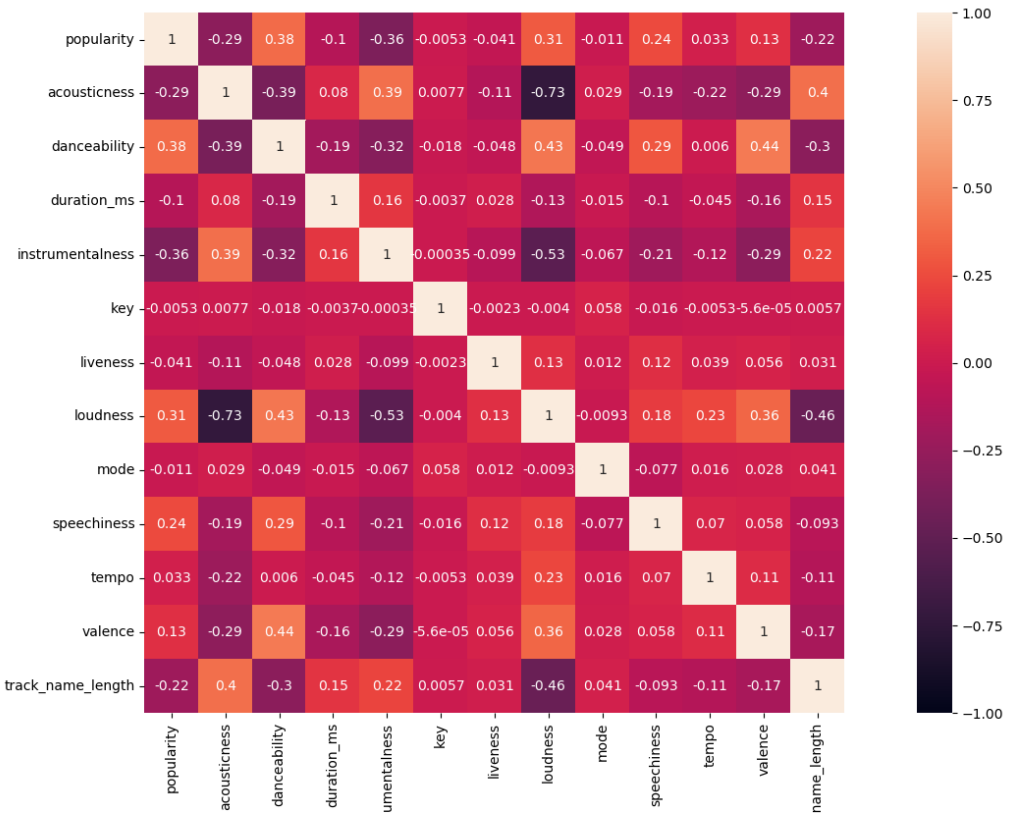

Figure 4: Matriz de Correlação

Através da imagem observada, conseguimos ver a correlação entre as diferentes *features*.

Feito este tratamento, passamos à utilização de vários modelos de aprendizagem de forma a prever os diferente estilos musicais e que melhor será explicado de seguido.

2.1.4 Descrição dos modelos desenvolvidos, quais as suas características, como e sobre que parâmetros foi realizado o *tuning* do modelo, características do treino, entre outros detalhes que seja oportuno fornecer

Na fase de treino do nosso *dataset*, os modelos de *MachineLearning* utilizados foram os lecionados nas aulas práticas e cuja implementação poderia vir a produzir bons resultados. De seguida iremos apresentar a descrição e características dos vários modelos desenvolvidos em casa *dataset* assim como o *tuning* de parâmetros realizado em cada modelo.

Decision Tree: Este algoritmo de aprendizagem é baseado em arvores de decisão e através da recursividade dos seus nodos. De seguida encontra-se o código feito para este modelo.

```
def decisionTree(x,y):
clf = DecisionTreeClassifier(random_state=2022)

scores = cross_val_score(clf,x,y,cv=10)
# scores = cross_val_score(clf, x, y,  cv=10)
print(scores.mean())
print(scores)

X_train,X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=
clf.fit(X_train,y_train)

predictions = clf.predict(X_test)
print(accuracy_score(y_test, predictions))
```

Random Forest: Este modelo de *EmsembleLearning* foi dos que produziu melhores resultados e também é baseado em Árvores de decisão. Ao contrário do algoritmo anterior, este não se baseia numa só decisão mas

sim na maioria decidida nas várias árvores criadas. Através da inserção de um fator aleatório, conseguimos alcançar maior diversidade e maior precisão. De seguida encontra-se a implementação do mesmo.

```
def randomForest(x,y):
clf = RandomForestClassifier(random_state=2022, n_estimators=100)

#scores = cross_val_score(clf,x,y.values.ravel(),cv=5)
#print(scores.mean())

predictions = cross_val_predict(clf,x,y.values.ravel(),cv=5)
print(accuracy_score(y, predictions))
precision = precision_score(y, predictions, average=None)
print(precision)
```

Vector Machine: Este algoritmo baseia-se em encontrar a melhor linha ou hiperplano que separa as amostras de dados em duas ou mais classes, com o intuito de maximizar a margem entre as classes. de seguida expomos o codigo referente a sua implementação.

```
def vectorMachine(x,y):
clf = SVC(random_state=2022)
#scores = cross_val_score(clf,x,y.values.ravel(),cv=5)
#print(scores.mean())
#print(scores)

X_train,X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=
clf.fit(X_train,y_train.values.ravel())
predictions = clf.predict(X_test)
print(accuracy_score(y_test, predictions))
```

Logistic Regression: Este algoritmo utiliza uma função logística para criar um modelo capaz de prever uma variável através de várias dependentes. De seguida, encontra-se ilustrado o código deste modelo. Não foi dos modelos que produziu melhores resultados.

```
def logisticRegression(x,y):
clf = LogisticRegression(random_state=2022,solver='liblinear',max_iter=1000)
scores = cross_val_score(clf,x,y.values.ravel(),cv=10)
print(scores.mean())
```

2.1.5 Sumário dos resultados obtidos e respetiva análise crítica

De seguida, mostramos uma tabela com os resultados obtidos para os diferentes modelos de exploração de *Machine Learning*.

Table 1

Modelos	
modelo de exploração	***Accuracy***
DecisionTree	0,6345
RandomForest	0,7443
VectorMachine	0,2779
LogisticRegression	0.4799

Podemos verificar que fazemos uma previsão boa para as colunas de *Rap*, *Classical* e *Anime*, nestas a accuracy chega a ser 80% ou perto. Há uma dificuldade em prever as colunas de *Country* e *Eletronic*, esta dificuldade passa pela previsão incorreta de *Contry* com *Rock* e *Eletronic* com *Jazz/Blues*.

2.1.6 Apresentação de sugestões e recomendações após análise dos resultados obtidos e dos modelos desenvolvidos

Por fim, a última etapa consiste em analisar os resultados obtidos no contexto do problema dado, bem como a sua resposta. Por isso, iremos apresentar, do nosso ponto de vista, o tratamento de dados e respetivo modelo

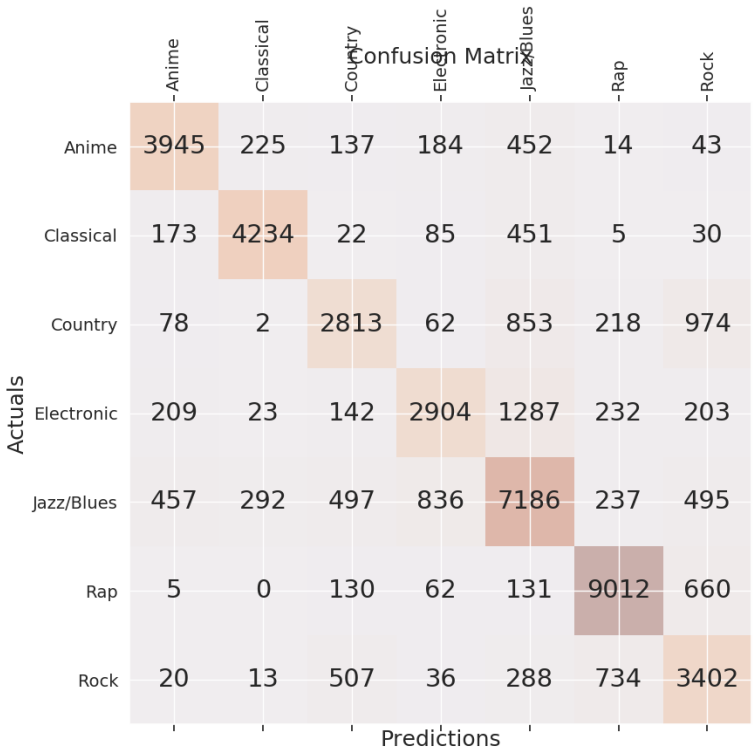

Figure 5: Caption

que melhor responderam a cada um dos problemas.

Assim sendo, a partir da tabela anteriormente apresentada, concluímos que o melhor resultado obtido foi pelo modelo do *RandomForest*.

O modelo de $VectorMachine$ tem um resultado mau, o que era de esperar, visto que o número de features é bastante reduzido comparando com o número de entradas.

O modelo de Decision Tree não apresenta resultados tão bons quanto o RandomForest, uma vez que as árvores de decisão tem em conta todas as características do conjunto de dados, até mesmo as mais irrelevantes, gerando então um modelo com muita capacidade de memorização e pouca capacidade de generalização. Por fim, o modelo *LogisticRegression* apresenta resultados baixos porque o dataset escolhido é um problema de classificação mais complexo, uma vez que os dados tem múltiplas classes e/ou uma relação não linear entre as características e a variável alvo.

Após a análise dos resultados obtidos, uma das sugestões seria a implementação de mais modelos como o XGBoost e adaBoost, visto que são modelos de Ensemble Learning projetados para aumentar a precisão do modelo final, bem como ajudam a reduzir o overfitting e diminuir o erro.

2.2 DATASET COMPETIÇÃO

2.2.1 Quais os domínios a tratar, quais os objetivos e como se propõem a atingi-los

Relativamente a este dataset, disponibilizado pela equipa docente, é constituido por 13 colunas, com o objetivo de prever o número de incidentes rodoviários que irão acontecer na cidade de Guimarães a uma determinada hora. Assim sendo, o *dataset* é dividido em dois: de aprendizagem e de teste. 5000 e 1200

No primeiro dataset, este possui 5000 entradas e vai ser sobre este que iremos treinar os diferentes modelos de aprendizagem de forma a prever

Já o dataset de teste tem o intuito de validar a accuracy do modelo em dados ainda não vistos pelo mesmo, nao sendo fornecida o *target* referente à quantidade de incidentes rodoviários. Ademais, concluímos que o dataset dado é trata-se de um modelo de classificação.

De seguida iremos explicar quais os modelos de aprendizagem supervisionada que foram aplicados e qual demonstrou melhor precisão nos seus resultados.

2.2.2 Qual a metodologia seguida e como foi aplicada

A metodologia para o tratamento deste dataset foi igual ao anterior, ou seja, utilizou-se a metodologia CRISP-DM.

2.2.3 Descrição e exploração detalhada de ambos os datasets e de todo e qualquer tratamento efetuado

De seguida, observamos quais *features* teriam valor único, uma vez que aqueles que o apresentam são irrelevante estar presente no dataset, porque não tem qualquer influência para o resultado final. Com isto, retiramos as features que apresentam apenas um único valor.

```
print(df.nunique(axis=0))
df=df.drop(     ['city_name', 'avg_precipitation'], axis=1)
test=test.drop(['city_name', 'avg_precipitation'], axis=1)
```

Posteriormente, elaboramos um *parse* com o intuito de saber quais eram os diferentes tipos de estrada que seriam afetadas, através da inicial.

```
    roads = {}

def parse_roads(x,letter = ''):
    if type(x)!=str:
        return 0
    if x==',':strip():
        return 0
    x = x.replace(' ','')
    mylist = list(x.split(','))
    if '' in mylist:
        mylist.remove('')
    if letter == '':
        return len(mylist)
    res = list(map(lambda x: x.strip().startswith(letter),mylist))
    res = res.count(True)
    return 1 if res > 0 else 0
```

Assim sendo, este *parse* elaborado serviu para decompor a feature *affect_roads* em 4 tipos diferentes, podendo os vaores ser 1 ou 0, respetivamente:

- affected_roads_N: referentes estradas nacionais
- affected_roads_I: referentes itenerários
- affected_roads_R: referentes estradas regionais
- affected_roads_E: referentes estradas regionais

Para além disso, alteramos *affect_roads* para dizer o número de estradas afetadas:

```
df['affected_roads'] = df["affected_roads"].apply(lambda x : parse_roads(x))
```

Passamos a features delay de categórico para numérico, com a seguinte função:

```
def parse_delay(x):
    if x == 'UNDEFINED':
        return 0
     elif x== 'MODERATE':
        return 1
     else:
        return 2

df['magnitude_of_delay'] = df['magnitude_of_delay'].apply(parse_delay)
```

Assim como a feature avg_rain:

```
    def parse_rain(x):
    if x == 'Sem Chuva':
        return 0
    elif x== 'chuva fraca':
        return 1
    elif x== 'chuva moderada':
        return 2
    else:
         return 3

 df['avg_rain'] = df['avg_rain'].apply(parse_rain)
```

E luminosity:

```
 def parse_luminosity(x):
    if x == 'DARK':
        return 0
    elif x== 'LOW_LIGHT':
        return 1
    else:
        return 2

df['luminosity'] = df['luminosity'].apply(parse_luminosity)
```

Depois das alterações efetuadas anteriormente, aplicamos a elaboração de uma matriz de correlação para ver a diferente correlação entre as diferentes features.

Para além das alterações já referidas anteriormente, para a *feature record_ date* dividimos a data em dia, mês, ano e hora, com o intuito de obter mais informação concreta acerca da data. No fim, eliminamos o *feature* original para não existir dados duplicados

Também, verificamos se existe duplicados e valores únicos para se eliminar as features na eventualidade de haver.

Por fim, elaboramos e aplicamos um parse com intuito de passar de numérico para categórico referente ao *incidents*, passando a exitir a feature *parse_ incidents* em vez de *incidents*.

```
    def parse_incidentes(x):
   if x == 'None':
       return 0
   elif x== 'Low':
       return 1
   elif x== 'Medium':
       return 2
   elif x== 'High':
       return 3
   elif x== 'Very_High':
       return 4
```

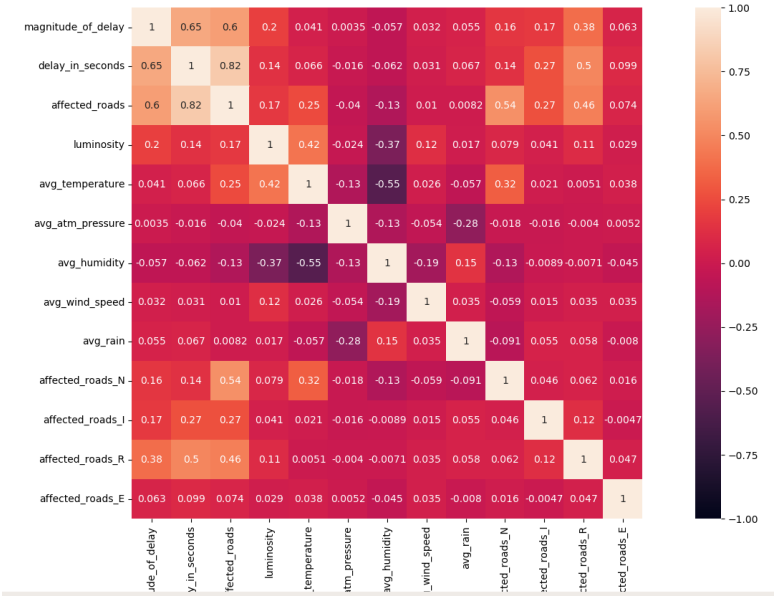

Figure 6: Matriz de Correlação

```
df['incidents'] = df['incidents'].apply(parse_incidentes)
```

2.2.4 Descrição dos modelos desenvolvidos, quais as suas características, como e sobre que parâmetros foi realizado o tuning do modelo, características do treino, entre outros detalhes que seja oportuno fornecer

Em relação aos modelos utilizados para o tratamento deste dataset, todos os que foram abordados no dataset de grupo foram implementados para este. Alguns destes foram eliminados uma vez que apresentavam pior *accuracy* e quando utilizados nas submissões não era viáveis.

2.2.5 Sumário dos resultados obtidos e respetiva análise crítica

No dataset de treino o resultado obtido no Kaggle é :

Figure 7: Resultado obtido com aproximadamente 30% dos dados de teste

Enquanto que o dataset de teste o resultado obtido é:

Figure 8: Resultado obtido com aproximadamente 70% dos dados de teste

Através dos resultados acima apresentados e pelas várias submissões feitas, conseguimos concluir que o melhor modelo de aprendizagem foi o Random Forest.

2.2.6 Apresentação de sugestões e recomendações após análise dos resultados obtidos e dos modelos desenvolvidos

Através da plataforma Kaggle, foram feitas várias submissões de forma a verificar a accuracy dos vários modelos de treino utilizados. Conforme estes resultados, fomos aplicando diferentes modelos e um melhor tratamento de dados de forma a subir na competição.

A primeira tentativa de submissão com Decision Tree obtivemos os seguintes resultados no público e no privado.

Figure 9: Primeira Tentativa com Decision Tree

De seguida utilizamos o Random Forest e obtivemos melhores resultados de previsão.

Figure 10: Primeira Tentativa com Random Forest

Após realizarmos o melhor tratamento de dados para o modelo acima referido, a nossa submissão final e melhor foi:

Figure 11: Submissão Final com Random Forest

3 Conclusão

Nesta fase final do projeto constatamos que conseguimos cumprir com tudo o que nos foi pedido.

Consideramos que foi um projeto bastante interessante e desafiante, visto que nos obrigou a interligar conceitos e matéria lecionados nas aulas teóricas com a componente mais prática que nos foi instruída ao longo das aulas PLs, nomeadamente, na conceção de modelos de aprendizagem.

Numa fase inicial, sentimos um pouco de dificuldade em colocar em prática o que fomos aprendendo ao longo do semestre, mas com o empenho e dedicação de todos os elementos conseguimos superar as nossas dificuldades.

Assim, enquanto grupo conseguimos distribuir bem o trabalho entre todos. Ajudamo-nos mutuamente e, de forma geral, tivemos um aproveitamento positivo.

Concluindo, este projeto ajudou-nos a desenvolver novas aptidões e a consolidar toda a matéria lecionada em aula.