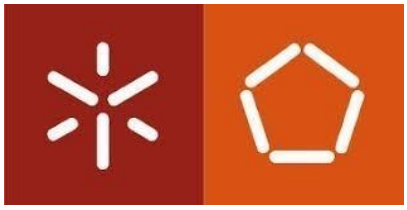
Universidade do Minho

Mestrado em Engenharia Informática



Dados e Aprendizagem Automática
1.º Ano, 1.º Semestre
Ano letivo 2021/2022

Trabalho Prático
Janeiro 2021

Grupo 52
Luis Pereira – PG47424

Introdução

Este trabalho prático foi desenvolvido no âmbito da UC de **Dados e Aprendizagem Automática** e tem como principal objetivo desenvolver e implementar modelos **Machine Learning**.

Para tal foram desenvolvidos vários modelos de Machine learning de maneira a responder a dois datasets: um para previsão do **fluxo de tráfego rodoviário na cidade do porto** e outro sobre **previsão de doenças**.

Neste relatório são apresentadas todas as etapas da realização deste modelos, desde a análise inicial dos dados de cada dataset, o processamento, o tuning, a validade até aos testes de cada modelo. De modo a concluir este trabalho iremos também realizar uma análise crítica dos resultados obtidos para cada modelo.

Dataset Tráfego Rodoviário

Análise dos dados

Este dataset inclui os seguintes atributos:

- **city_name** - nome da cidade em causa;
- **record_date** - o timestamp associado ao registo;

- **average_speed_diff** - a diferença de velocidade corresponde à diferença entre (1.) a velocidade máxima que os carros podem atingir em cenários sem trânsito e (2.) a velocidade que realmente se verifica. Quanto mais alto o valor, maior é a diferença entre o que se está a andar no momento e o que se deveria estar a andar sem trânsito, i.e., valores altos deste atributo implicam que se está a andar mais devagar;

- **average_free_flow_speed** - o valor médio da velocidade máxima que os carros podem atingir em cenários sem trânsito;

- **average_time_diff** - o valor médio da diferença do tempo que se demora a percorrer um determinado conjunto de ruas. Quanto mais alto o valor maior é a diferença entre o tempo que demora para se percorrer as ruas e o que se deveria demorar sem trânsito, i.e., valores altos implicam que se está a demorar mais tempo a atravessar o conjunto de ruas;

- **average_free_flow_time** - o valor médio do tempo que demora a percorrer um determinado conjunto de ruas quando não há trânsito;

- **luminosity** - o nível de luminosidade que se verificava na cidade do Porto;

- **average_temperature** - o valor médio da temperatura para o record_date na cidade do Porto;
- **average_atmosp_pressure** - o valor médio da pressão atmosférica para o record_date;
- **average_humidity** - o valor médio da humidade para o record_date;
- **average_wind_speed** - o valor médio da velocidade do vento para o record_date;
- **average_cloudiness** - o valor médio da percentagem de nuvens para o record_date;
- **average_precipitation** - o valor médio de precipitação para o record_date;
- **average_rain** - avaliação qualitativa da precipitação para o record_date.

O objectivo da análise deste dataset tem como intuito ser capaz de prever a intensidade do trânsito num dado momento. Para tal o atributo respectivo para previsão é **“average_speed_diff”.**

Common Values

Value	Count	Frequency (%)
None	2200	32.3%
Medium	1651	24.2%
Low	1419	20.8%
High	1063	15.6%
Very_High	479	7.0%

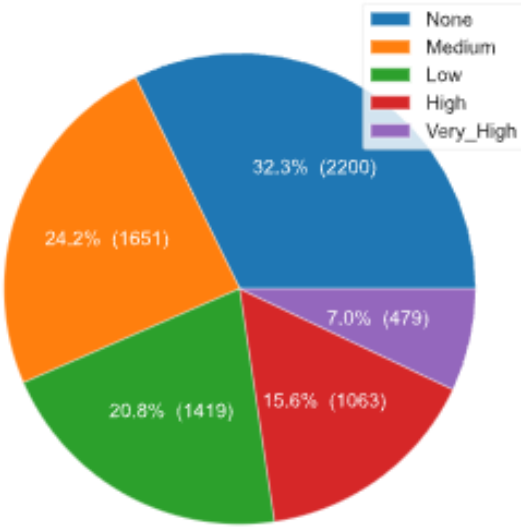


Figura 1: Distribuição do atributo **average_speed_diff**

Como se pode observar pela figura anterior, o atributo é composto pelos parâmetros **"Low"**, **"Medium"**, **"High"**, **"Very High"** e **"None"** que definem a intensidade do trânsito.

Através do gráfico é possível elaborar uma ideia geral do trânsito existente no porto. Em que durante a maior parte do dia ou não existe tráfico ou é relativamente baixo a médio. Como podemos observar "High e "Very High" tráfego somente representa 15.6% e 7%

Também já se consegue prever que poderemos usar técnicas de Oversampling para balancear o nosso atributo, tendo em atenção Overfitting.

Pre-Processing

Numa fase inicial da construção deste modelo de Machine Learning, foi criado um jupyter notebook para realizar o tratamento de dados e uma análise crítica do conjunto de dados fornecidos nos datasets e determinou-se quais seriam os dados a serem utilizados, quais teriam que ser alterados e quais não seriam utilizados.

Inicialmente foi utilizada a ferramenta de **ProfileReport** do pandas para gerar um relatório detalhado do dataset de treino e teste. Com base neste relatório, foi de imediato percebido que existem features com valores constantes e em falta.

Valores Constantes

Em particular foram encontradas duas features que apresentavam valores constantes para todas as suas linhas, **city_name e average_precipitation**. Por estas features serem constantes não vão ter importância para o treino do nosso modelo logo foram retiradas.

Valores em falta

Foram encontradas duas features com valores em falta, **average_cloudiness** e **average_rain**, com respectivamente 39.4% e 91.7% de valores em falta.

Após uma observação destas features foi aplicada moda à **average_cloudiness** e verificou-se que não existia que em **average_rain** não existia categoria para identificar quando não chovia pelo que assumi que os valores em falta seriam dias em que não havia chuva. Representando dias com chuva com 1 e dias sem com 0.

Extended Features

Foram criadas novas features a partir das existentes. Da feature **record_date** foi inicialmente expandida em:

- Hour
- Day
- WeekDay
- Is_Weekend
- Month
- Year

A partir da fórmula da velocidade: $Speed = \frac{Distance}{Time}$

- **AVERAGE_DISTANCE_FREE** - Distância percorrida sem trânsito
- **AVERAGE_TIME -** Tempo médio que demorou a percorrer a distância
- **AVERAGE_SPEED -** Velocidade média a que a viatura tem que se deslocar para percorrer certa distância em certo tempo.
- **AVERAGE_SPEED_DIFF_Num -** Diferença numérica entre a velocidade média e a velocidade sem trânsito.

One-Hot Encoding e Transforms

Foi aplicado um **CyclicalTransformer** às features cíclicas, **Hour**, **Day**, **Weekday**, **Month**, **Year** de forma a manter a sua forma cíclica, resultando na criação de 2 colunas sin e cos para cada feature.

Para efetuar o **encoding** das features categóricas **Luminosity** e **AVERAGE_CLOUDINESS** foi utilizada a função **get_dummies** do pandas, criando número de colunas igual a quantidade de categorias. Para manter uma quantidade mais controlada de features foram analisadas e agrupadas categorias.

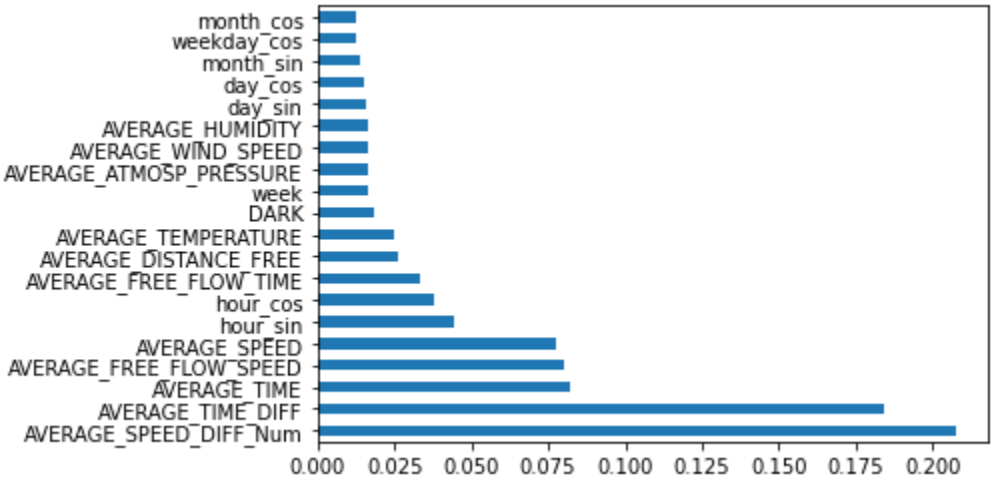


Figura 2: Importância das features

Model Training

Foram utilizadas três alternativas de **Machine Learning**. A primeira em que sampling, algoritmos, tuning e validação são todos efetuados e chamados manualmente. Uma segunda alternativa em que é utilizada a ferramenta **AutoML Pycaret** que automatiza muito do processo de Machine Learning. Finalmente a terceira que tira partido de **Neural Networks**.

Manual

Inicialmente é efetuado o **Train_Test_Split** de 80%, 20% correspondente.

Inicializados e declaradas as variáveis para tanto as técnicas de sampling como algoritmos de previsão, utilizando sempre que possível uma seed random_state 2021.

Sampling

Técnicas de Sampling testadas:

- RandomOverSampler
- SMOTE
- ADASYN
- BorderlineSMOTE
- SMOTEENN
- SMOTETomek
- SVMSMOTE

Algoritmos

Algoritmos de Machine Learning testados:

- LogisticRegression
- RandomForestClassifier
- KNeighborsClassifier
- DecisionTreeClassifier
- AdaBoostClassifier
- SVC
- XGBClassifier
- RidgeClassifier
- LGBMClassifier

K-Fold Validation e Cross_Val_Score

De forma a reduzir o **Overfitting** introduzido pela quantidade de features e Oversampling foi inicialmente criado uma pipeline que aplica sampling somente ao X de training e depois o algoritmo ML. De forma a melhor validar o modelo foi introduzida ao **cross_val_score** uma estratégia de **StratifiedKFold** com 10 splits e shuffle com uma random state seed de 2021. O que significa que os dados serão baralhados e separados em 10 partes reduzindo assim o bias do modelo.

Tuning

Utilizando os scores do **Cross_Val_Score** como guia, foi efetuado um melhor tuning dos algoritmos com melhor accuracy. Começando por declarar uma grid de parâmetros para cada algoritmos e inicialmente correr um RandomizedSearchCV para encontrar em menor tempo os parâmetros aproximados do melhor, sendo após passados ao GridSearchCV e manualmente alterados até encontrar os melhores parâmetros possíveis para cada algoritmo.

Validação

A validação foi efetuada em 3 etapas. Primeiramente é criado um scatterplot entre os valores predicted e os valores reais, podendo ser visualizado a distribuição dos valores.

Em segundo lugar são criadas as Matrizes de confusão com coloração e valores em percentagem, deste modo podemos ter uma percepção rápida das categorias que estão a ter uma previsão melhor ou pior.

Por último foi utilizado **sklearn.metrics** e a ferramenta de **classification_report** de forma a gerar um relatório com os precision, recall e F1 scores para todos os diferentes algoritmos e samplings.

Exemplo Classification_Report

```
Pipeline(steps=[('borderlinesmote',
                 BorderlineSMOTE(kind='borderline-2', random_state=2021)),
                ('lgbmclassifier', LGBMClassifier(random_state=2021))])
```

Accuracy: 0.82

Micro Precision: 0.82
Micro Recall: 0.82
Micro F1-score: 0.82

Macro Precision: 0.81
Macro Recall: 0.82
Macro F1-score: 0.81

Weighted Precision: 0.82
Weighted Recall: 0.82
Weighted F1-score: 0.82

Classification Report

	precision	recall	f1-score	support
None	0.80	0.78	0.79	213
Low	0.72	0.74	0.73	284
Medium	0.83	0.78	0.80	330
High	0.89	0.90	0.90	440
Very_High	0.82	0.88	0.84	96
accuracy			0.82	1363
macro avg	0.81	0.82	0.81	1363
weighted avg	0.82	0.82	0.82	1363

Como podemos observar a average accuracy está por volta de 82%
Sendo a pior categoria **“LOW”** por volta dos 72% e seria a primeira categoria a olhar para melhorar o modelo.

Pycaret AutoML

Pycaret é uma libraria python **openSource** de **Auto Machine Learning** que visa aumentar a produtividade e diminuir a dificuldade de usabilidade devido a sua estratégia low code.

Com muito pouco código podem ser passados varios argumentos ao setup do pycaret. Como train_size, random seed, normalização, balanceamento e método.

```
classify = setup(data = training , target =
   'AVERAGE_SPEED_DIFF',train_size=0.8,session_id=2021,normalize =
   True,fix_imbalance=True,fix_imbalance_method=sm)
```

Pycaret conta também com funções de comparação de algoritmos de machine learning, ensemble, boosting e relatórios de validação de modelos.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8121	0.9649	0.8114	0.8150	0.8127	0.7549	0.7553
xgboost	Extreme Gradient Boosting	0.8106	0.9638	0.8076	0.8136	0.8112	0.7529	0.7533
rf	Random Forest Classifier	0.8053	0.9647	0.8090	0.8099	0.8058	0.7464	0.7473
gbc	Gradient Boosting Classifier	0.7985	0.9632	0.8026	0.8023	0.7989	0.7376	0.7383
et	Extra Trees Classifier	0.7976	0.9628	0.7985	0.7995	0.7971	0.7356	0.7363
lr	Logistic Regression	0.7651	0.9519	0.7729	0.7725	0.7644	0.6948	0.6964
lda	Linear Discriminant Analysis	0.7221	0.9263	0.7349	0.7206	0.7170	0.6376	0.6392
dt	Decision Tree Classifier	0.7201	0.8210	0.7175	0.7245	0.7210	0.6356	0.6362
nb	Naive Bayes	0.7130	0.9200	0.7211	0.7148	0.7072	0.6256	0.6280
svm	SVM - Linear Kernel	0.6711	0.0000	0.6747	0.6722	0.6594	0.5720	0.5767
knn	K Neighbors Classifier	0.6185	0.8486	0.6350	0.6231	0.6176	0.5046	0.5059
ridge	Ridge Classifier	0.6051	0.0000	0.6041	0.6008	0.5566	0.4789	0.4975
ada	Ada Boost Classifier	0.5054	0.7557	0.5141	0.5602	0.4717	0.3657	0.3873
qda	Quadratic Discriminant Analysis	0.4775	0.7699	0.5767	0.4086	0.4051	0.3475	0.3989
dummy	Dummy Classifier	0.2219	0.5000	0.2000	0.0517	0.0833	0.0000	0.0000

Figura 3: Comparação e validação de modelos Pycaret

Neural Networks

Inicialmente foram processados os dados, fazendo um encoding dos valores categóricos do atributo para valores de 0-4 e posteriormente aplicado um **MinMaxScaler** entre 0 a 1 as features.

Foi criada uma pequena função para calcular o peso de cada categoria a prever.

Recorrendo ao Keras tuner e após ser criado um esqueleto de modelo foram otimizados os hiperparâmetros, quantidade e profundidade dos nodos.

Validação

De modo a validar o modelo foram criados 2 plots da progressão da accuracy e model loss durante as várias **epochs**. Podendo imediatamente perceber se utilizou epochs a mais ou insuficientes. Foi criada também uma matriz de confusão e utilizada a função **evaluate** da library **Keras .**

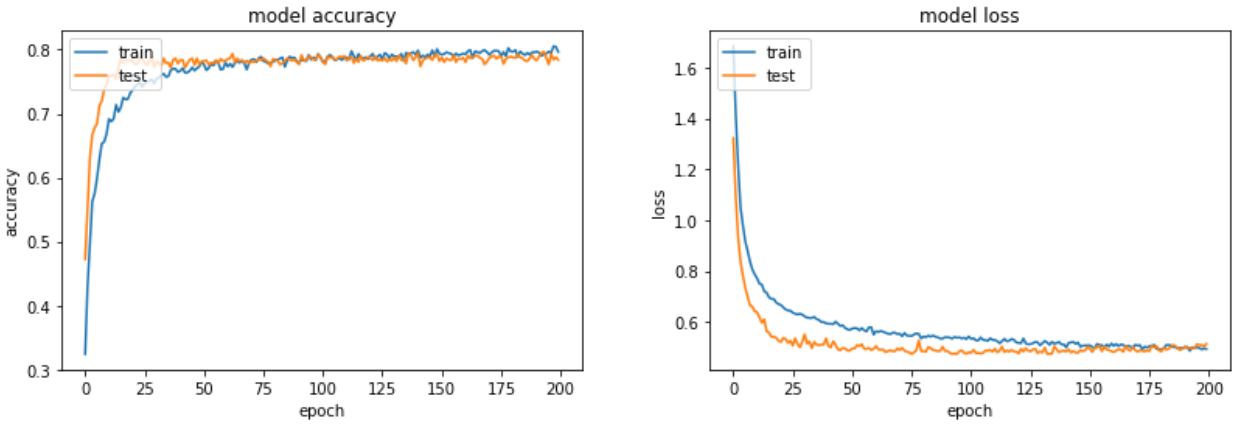


Figura 4: Neural Networks model Accuracy and Loss

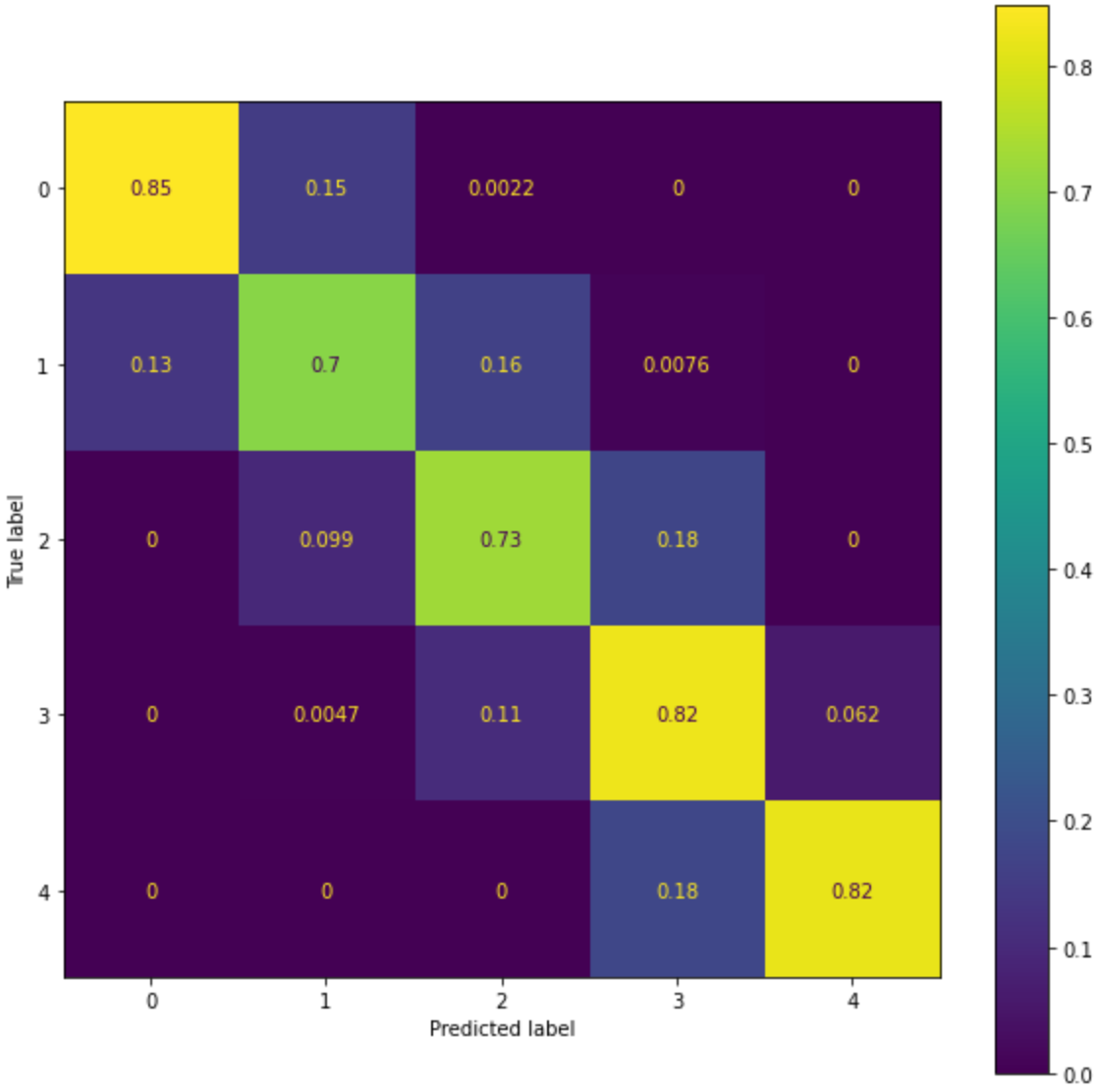


Figura 5: Neural Network model Confusion Matrix

Dataset Disease Prediction

Análise dos dados

Este dataset inclui vários sintomas de tipo binário e prognósticos categóricos.
Um total de 133 colunas e 4920 linhas.

Este dataset inclui os seguintes atributos:

- 132 Features de Sintomas
- 1 Atributo com o prognóstico

Pre-Processing

Numa fase inicial da construção deste modelo de Machine Learning, foi criado um jupyter notebook para realizar o tratamento de dados e uma análise crıtica do conjunto de dados fornecidos nos datasets e determinou-se quais seriam os dados a serem utilizados, quais teriam que ser alterados e quais não seriam utilizados.

Inicialmente foi utilizada a ferramenta de **ProfileReport** do pandas para gerar um relatório detalhado do dataset de treino e teste. Com base neste relatório, foi de imediato percebido que existem features com valores constantes e em falta.

Value	Count	Frequency (%)
Fungal infection	120	2.4%
Hepatitis C	120	2.4%
Hepatitis E	120	2.4%
Alcoholic hepatitis	120	2.4%
Tuberculosis	120	2.4%
Common Cold	120	2.4%
Pneumonia	120	2.4%
Dimorphic hemmorhoids(piles)	120	2.4%
Heart attack	120	2.4%
Varicose veins	120	2.4%
Other values (31)	3720	75.6%

Figura 6: Distribuição do atributo Prognosis

Uma coluna **“Unnamed: 133”** foi encontrada vazia no dataset o qual foi retirada.
A coluna fluid_overload tem valor constante 0 pelo qual foi também retirada do dataset.
Ficando assim com 130 features e um atributo.
Como podemos observar na figura acima todas as categorias do nosso atributo estão balanceadas, podendo então proceder a feature engineering e training.

	symptom	importance
104	polyuria	0.000986075722798965
15	weight_gain	0.000987439914246877
72	swollen_extremeties	0.0010161120209910737
125	silver_like_dusting	0.0018141672027199246
69	puffy_face_and_eyes	0.0018277525556415087
...	...	...
11	vomiting	0.022663116988338908
31	headache	0.02469541161235492
14	fatigue	0.024820974549335947
0	itching	0.025639053270424265
25	high_fever	0.028755129524351504

131 rows × 2 columns

Figura 7: Importância de cada feature

Manual

Inicialmente é efetuado o **Train_Test_Split** de 80%, 20% correspondente.
Inicializados e declaradas as variáveis para os algoritmos de previsão, utilizando sempre que possível uma seed random_state 2021.

Algoritmos

Algoritmos de Machine Learning testados:

- LogisticRegression
- RandomForestClassifier
- KNeighborsClassifier
- DecisionTreeClassifier
- AdaBoostClassifier
- SVC
- XGBClassifier
- RidgeClassifier
- LGBMClassifier

K-Fold Validation e Cross_Val_Score

De forma a melhor validar o modelo foi introduzida ao **cross_val_score** uma estratégia de **StratifiedKFold** com 10 splits e shuffle com uma random state seed de 2021. O que significa que os dados serão baralhados e separados em 10 partes reduzindo assim o bias do modelo.

Validação

A validação foi efetuada em 3 etapas. Primeiramente é criado um scatterplot entre os valores predicted e os valores reais, podendo ser visualizado a distribuição dos valores.
Em segundo lugar são criadas as Matrizes de confusão com coloração e valores em percentagem, deste modo podemos ter uma percepção rápida das categorias que estão a ter uma previsão melhor ou pior.
Por último foi utilizado **sklearn.metrics** e a ferramenta de **classification_report** de forma a gerar um relatório com os precision, recall e F1 scores para todos os diferentes algoritmos e samplings.

Exemplo Classification_Report

	precision	recall	f1-score	support
(vertigo) Paroymsal Positional Vertigo	1.00	1.00	1.00	1
Acne	1.00	1.00	1.00	1
Alcoholic hepatitis	1.00	1.00	1.00	1
Allergy	1.00	1.00	1.00	1
Arthritis	1.00	1.00	1.00	1
Diabetes	1.00	1.00	1.00	1
Dimorphic hemmorhoids(piles)	1.00	1.00	1.00	1
Hepatitis B	1.00	1.00	1.00	1
Hepatitis C	1.00	1.00	1.00	1
Hypertension	1.00	1.00	1.00	1
Hyperthyroidism	1.00	1.00	1.00	1
Hypoglycemia	1.00	1.00	1.00	1
Hypothyroidism	1.00	1.00	1.00	1
Impetigo	1.00	1.00	1.00	1
Jaundice	1.00	1.00	1.00	1
Malaria	1.00	1.00	1.00	1
Migraine	1.00	1.00	1.00	1
Osteoarthristis	1.00	1.00	1.00	1
Tuberculosis	1.00	1.00	1.00	1
Typhoid	1.00	1.00	1.00	1
Urinary tract infection	1.00	1.00	1.00	1
Varicose veins	1.00	1.00	1.00	1
hepatitis A	1.00	1.00	1.00	1
accuracy			1.00	42
macro avg	1.00	1.00	1.00	42
weighted avg	1.00	1.00	1.00	42

Train Accuracy: 1.0
Test Accuracy: 1.0

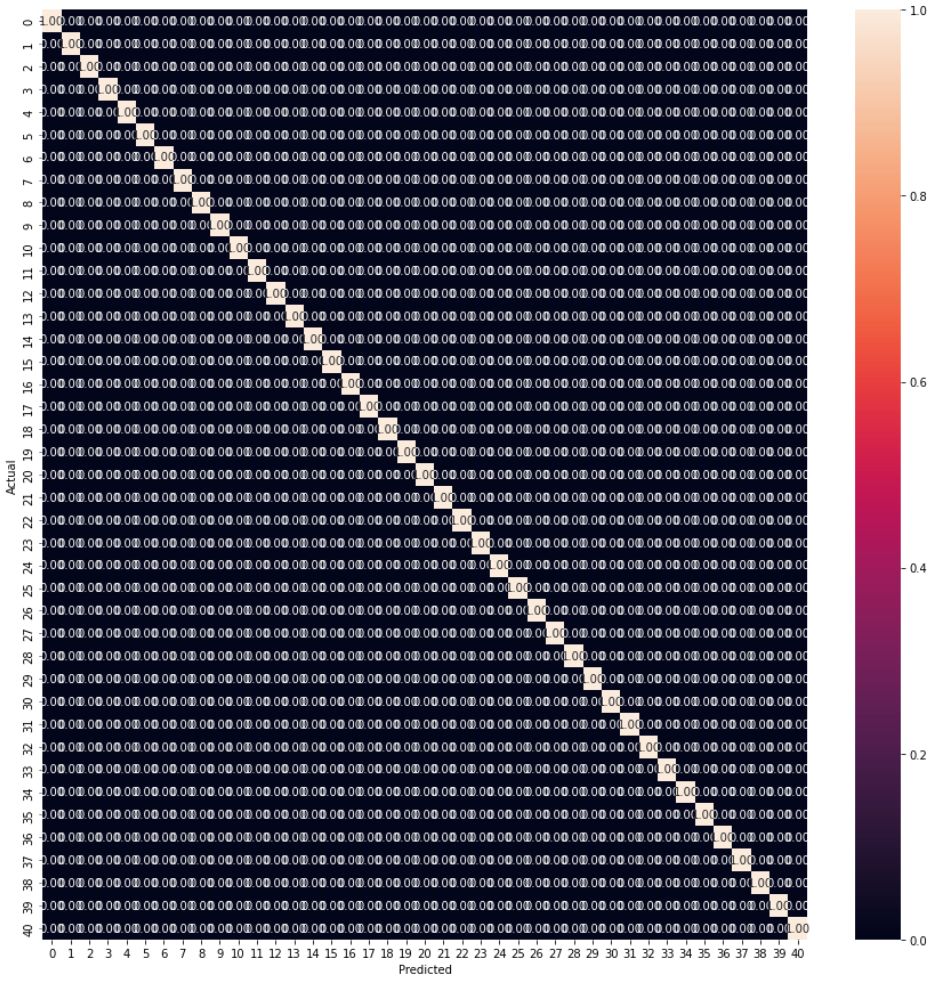


Figura 8: Matriz Confusão

Neural Networks

Foi inicialmente aplicado um **One_Hot_Encoding** via **get_dummies** do pandas para codificar o atributo de **"prognosis"**.

Após criado um modelo simples sequential foi adicionado um monitor de early stopping para caso a accuracy de validação não se altere durante 10 epochs. Não tendo assim o modelo de treinar por todas a epochs, parando quando não conseguir convergir mais.

```
early_stopping_monitor= EarlyStopping(patience=10,
   monitor='val_accuracy')
```

Validação

De modo a validar o modelo foram criados 2 plots da progressão da accuracy e model loss durante as várias **epochs**. Podendo imediatamente perceber se utilizou epochs a mais ou insuficientes. Foi criada também uma matriz de confusão e utilizada a função **evaluate** da library **Keras .**

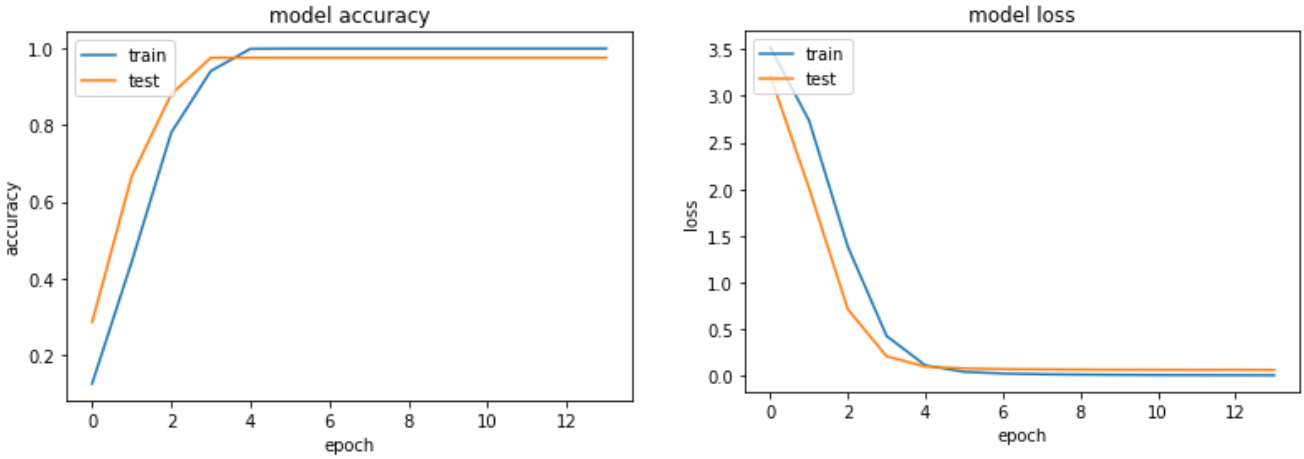


Figura 8: Neural Networks model Accuracy and Loss

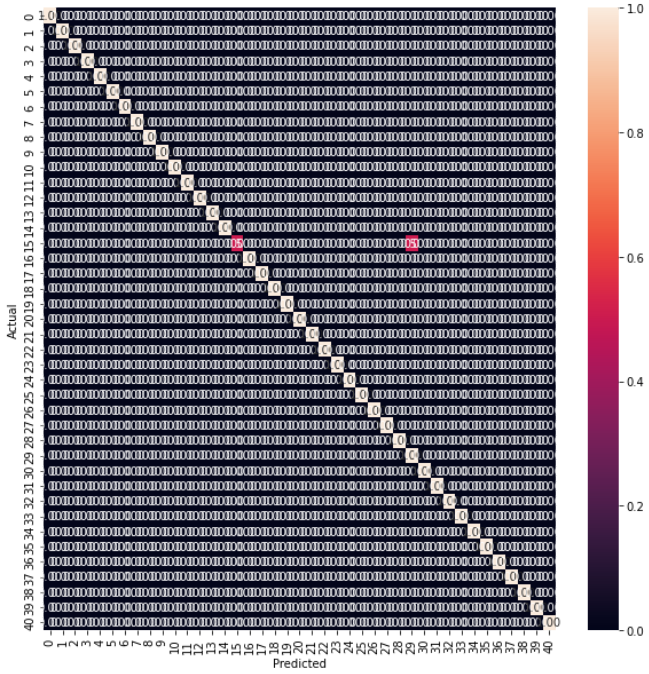


Figura 9: Matriz de confusão

Como podemos observar pelos resultados acima. Este dataset está bem equilibrado. Ao ponto que alguns modelos conseguem ter uma accuracy de previsão de 100%

Existe uma pequena melhoria que poderia ser feita com o hyperparametros do Keras tuner.

Conclusão

Ao longo deste trabalho, fui capaz de desenvolver várias competências nas áreas de data science, feature engineering, machine learning e criação de modelos.

Foram aplicadas várias técnicas de sampling, incluindo SMOTE, e experimentado com técnicas mistas com **Oversampling** e **Undersampling** em conjunção.

Verifiquei os problemas que o **Overfitting** traz ao nosso modelo assim como uma pobre limpeza e preparação dos dados.

Uma boa validação é necessária a cada passo do projeto de modo a verificar possíveis pontos de melhoria e hiperparâmetros a modificar.

Em conclusão podemos ver neste trabalho que o maior obstáculo a **Machine Learning** é o estado dos nossos dados. Existe uma diferença enorme entre o **Dataset Tráfego Rodoviário** e o **Dataset Disease Prediction**. Em que o Dataset de Tráfego é mais realista na qualidade dos dados que aparecem num problema de **ML**.