Universidade Federal de Pernambuco
Centro de Informática

Acitivy Recognition

Grupo:
Karl Sousa
Maria Eugênia
Mateus Silva

Professor: Leandro Maciel Almeida

1 Introdução

O processo de mineração de dados pode ser dividido em várias etapas, desde uma análise exploratória inicial para se conhecer os dados, até se chegar de fato na geração de novas informações a respeito do problema. Um modelo bastante utilizado é o CRISP-DM (*Cross Industry Standard Process for Data Mining*), cujo diagrama é apresentado na Figura 1.

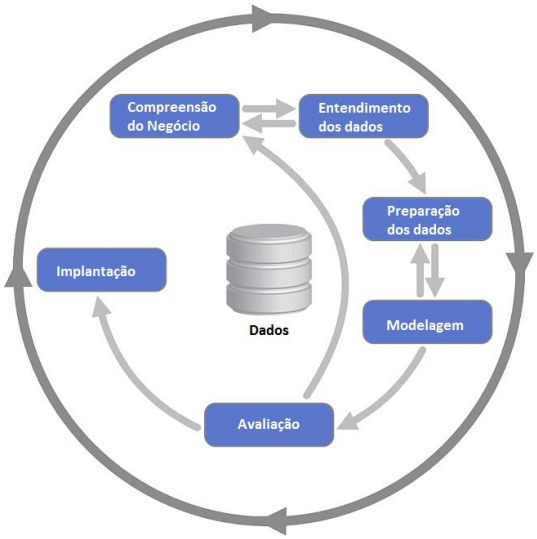


Figura 1: Diagrama com etapas do CRISP-DM.

Este relatório segue o processo de mineração de dados da base "Activity recognition with healthy older people using a batteryless wearable sensor", disponível no *UCI Machine Learning Repository*. A base é composta de oito atributos coletados por meio de um sensor passivo, que mede a aceleração em três eixos. A partir de antenas fixadas em uma sala, são captados esses sinais, como apresentado na Figura 2.

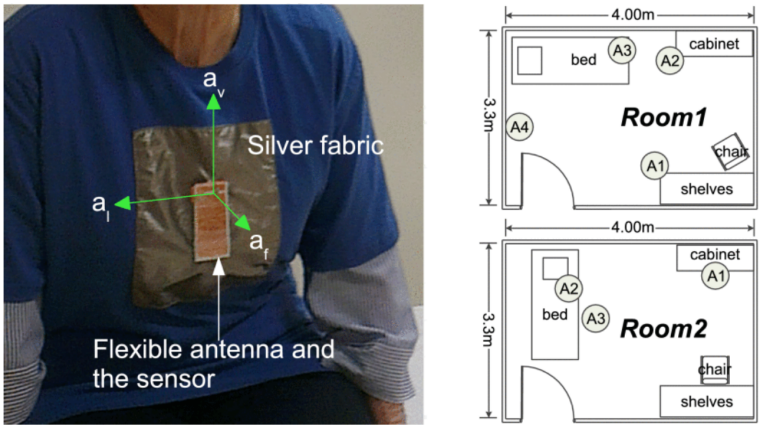


Figura 2: Equipamento utilizado para fazer medições nos idosos, e configurações das salas.

Como parte final para o processo de predição da classe dos exemplos, são utilizados os seguintes algoritmos de classificação:

- Árvore de decisão;
- k-*Nearest Neighbors* (kNN);
- Multilayer Perceptron (MLP);
- *Random Forest* (comitê de árvores de decisão);
- Comitê de MLP;
- Comitê heterogêneo, formado por árvore de decisão, kNN e MLP.

A linguagem de programação utilizada para desenvolver todas as etapas foi Python 3. O código do projeto pode ser encontrado neste link.

2 Análise exploratória

Por meio de análise estatística descritiva e visualização dos dados, pôde-se ter informações sobre as características da base de dados, e as possíveis dificuldades que poderiam ser encontradas na mineração dos dados. Um claro desafio está relacionado ao desbalanceamento dos dados. O número de exemplos por classe, está apresentado na Figura 3.

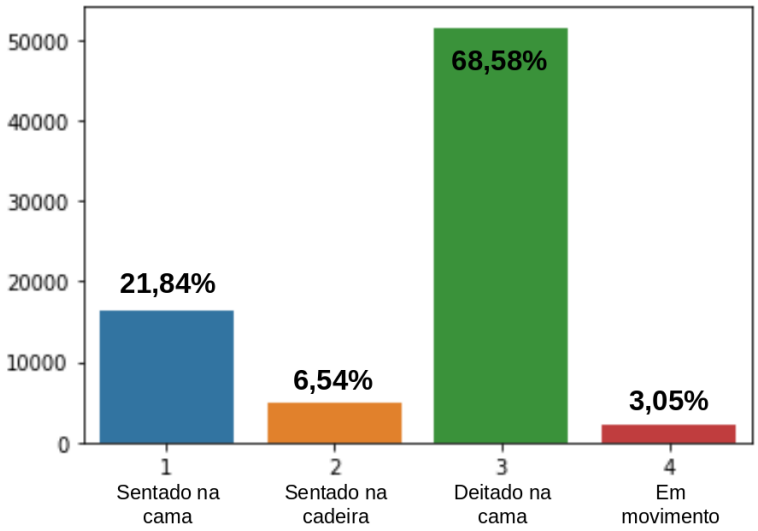


Figura 3: Distribuição de classes.

Na Figura 4 vemos o diagrama de caixa (*box plot*) para cada coluna da base de dados, onde as distribuições com maior número de *outliers* são a acaleração lateral, o RSSI e o tempo. Os atributos com comportamento mais próximo de uma distribuição normal são a frequência e a fase, apesar de isso não indicar necessariamente uma maior contribuição para a decisão do classificador.

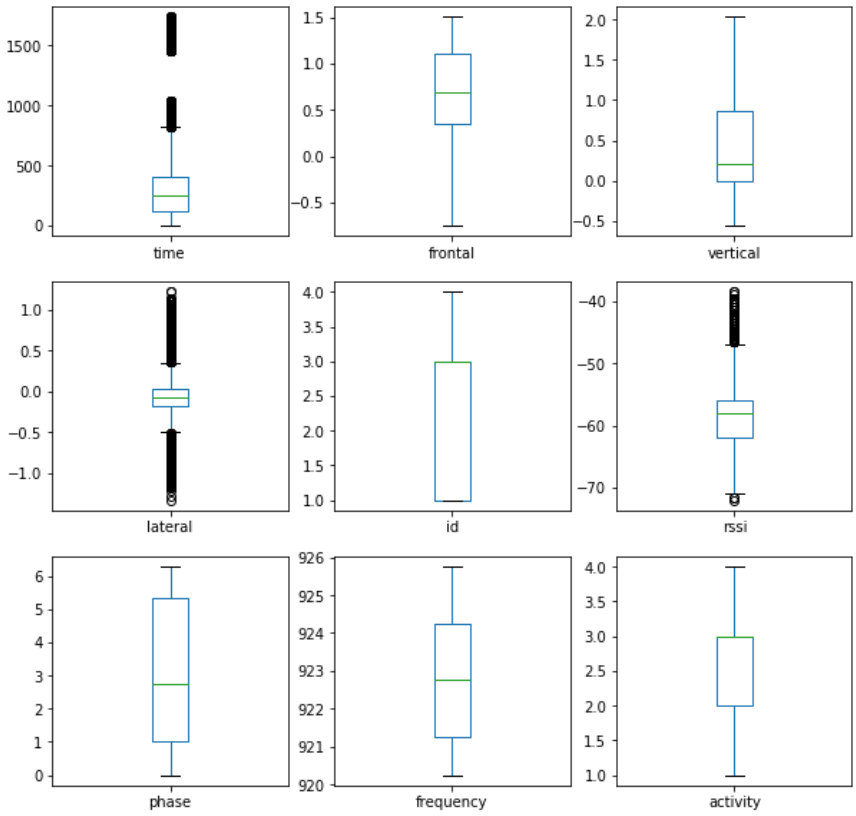
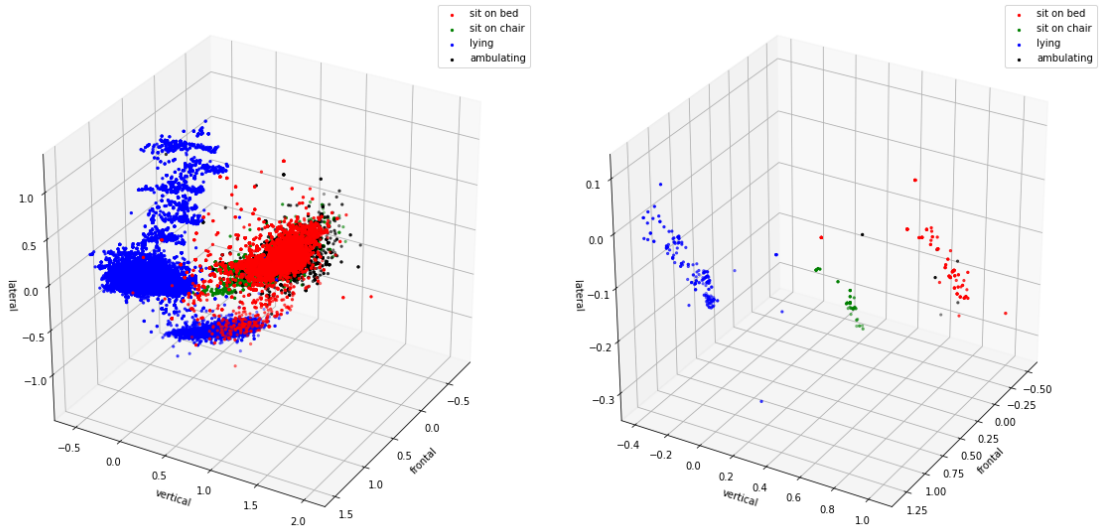


Figura 4: *Boxplots* dos atributos para o conjunto geral dos dados.

Para uma melhor análise visual da separabilidade dos dados, é apresentado na Figura 5 as acelerações nos três eixos, com classificação por cor. Vê-se que para os dados de um único ensaio, a separação é muito clara, com exceção da classe 4 (em movimento).



(a) Agrupamento de dados de todos os ensaios

(b) Dados de um único ensaio

Figura 5: Gráfico dos três eixos de aceleração, coloridos por classe.

3 Pré-processamento

No processo de preparação e pré-processamento dos dados, as subetapas consistiram em:

- Agrupamento dos dados, que estão separados por arquivos e pastas diferentes;
- Seleção de atributos (*feature engineering*);
- Normalização dos dados, para reescalonamento da extensão de cada atributo;
- Separação dos conjuntos de dados de treino e teste.

A base de dados consiste em vários arquivos com dados tabulares, em que cada arquivo representa o ensaio de coleta de dados de um idoso em uma sala equipada com antenas. A visualização inicial dos dados do *data-frame* pode ser vista a seguir:

	tempo	frontal	vertical	lateral	antena	rssi	fase	frequencia	atividade
0	0.00	0.27203	1.00820	-0.082102	1	-63.5	2.4252	924.25	1
1	0.50	0.27203	1.00820	-0.082102	1	-63.0	4.7369	921.75	1
2	1.50	0.44791	0.91636	-0.013684	1	-63.5	3.0311	923.75	1
3	1.75	0.44791	0.91636	-0.013684	1	-63.0	2.0371	921.25	1
4	2.50	0.34238	0.96229	-0.059296	1	-63.5	5.8920	920.25	1

No próprio nome do arquivo, está identificado se a pessoa em questão é homem ou mulher, e também em qual configuração de sala foi feita a coleta (S1 ou S2). Logo, algumas informações relevantes que estão presentes no próprio nome do arquivo não estão presentes no conteúdo interno do arquivo (tabela). A base de dados considerada para o problema, foi feita agrupando os dados de todos esses arquivos. Dessa forma, foi capturada essa identificação do sexo e da sala, e adicionados como novos atributos nos dados.

Feito isso, para separar todos os atributos, seleciona-se as colunas do *dataframe*, com exceção da última. A classe está localizada na última coluna.

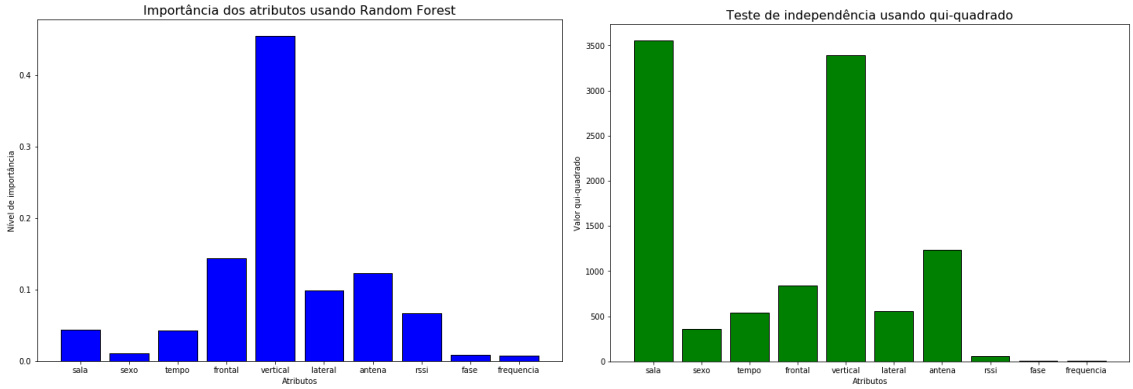
```
X = df.values[:, 0:-1]
y = df.values[:, -1]
```

Quanto à seleção de atributos, foram analisadas as correlação entre os atributos e a importância para a classificação da atividade. Essa etapa é de extrema importância, pois alguns algoritmos de classificação são prejudicados com a presença de atributos bastante correlacionados. Para a seleção dos atributos, foram feitas duas abordagens: por meio do *Random Forest* e do teste qui-quadrado, como pode ser visto na Figura 6. Apesar dos métodos atribuírem importâncias diferentes para cada atributo, ambos concordaram que os atributos frequência e fase não são relevantes ou são independentes da tarefa de classificar a atividade. Logo, esses atributos foram retirados e não serão utilizados.

Na normalização dos dados, para cada atributo é feito um reescalonamento da faixa de valores, de forma que a escala de cada atributo não tenha relevância na decisão para a classe a qual o exemplo pertence. Por meio do *MinMaxScaler*, é especificado a variação de valores dos dados, e nesse caso foi escolhida a faixa de zero a um.

Após realizados os passos de seleção de atributos (adição de dois atributos e exclusão de frequência e fase) e normalização, as cinco primeiras linhas do *dataframe* são vistas a seguir:

	sala	sexo	tempo	frontal	vertical	lateral	antena	rssi	atividade
0	0	0	0.160849	0.881062	0.168890	0.481822	0.000000	0.432836	3



(a) Random Forest (b) Teste qui-quadrado

Figura 6: Seleção de atributos segundo nível de importância.

1	1	1	0.292244	0.383782	0.324447	0.090887	0.333333	0.701493	3
2	1	1	0.435208	0.427024	0.306670	0.104520	0.333333	0.641791	3
3	0	1	0.023140	0.859441	0.235558	0.704556	0.000000	0.388060	3
4	0	0	0.139703	0.751351	0.235558	0.477276	0.666667	0.656716	3

Finalmente, é importante destacar a separação do conjunto de dados em treinamento e teste. Este, será guardado para avaliação final do melhor algoritmo encontrado, enquanto que os dados de treino serão divididos em treino e validação para ajuste de parâmetros e hiper-parâmetros. Na Figura 7 é apresentada essa separação.

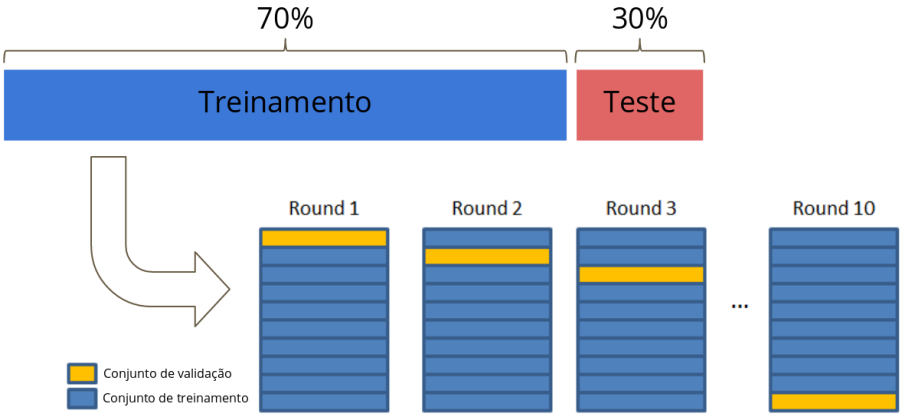


Figura 7: Separação dos dados por *holdout* entre treino e teste, e validação cruzada 10-fold para treinamento e validação.

4 Algoritmos de classificação

4.1 Árvore de decisão

O parâmetro avaliado foi a profundidade da árvore, em que pode se definir um valor máximo, evitando um possível *overfit* por parte do classi-

ficador. Treinando esse classificador sem impor um limite, a profundidade da árvore criada foi de 30. Dessa forma, foram criados gráficos (Figura 8) para as métricas de acurácia, F1 score e MCC, variando-se o parâmetro *max_depth* de 4 a 28. O melhor valor considerado foi então *max_depth=12*.

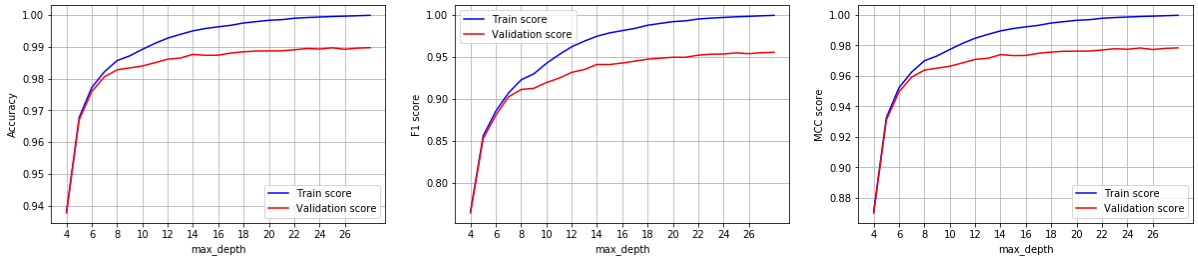
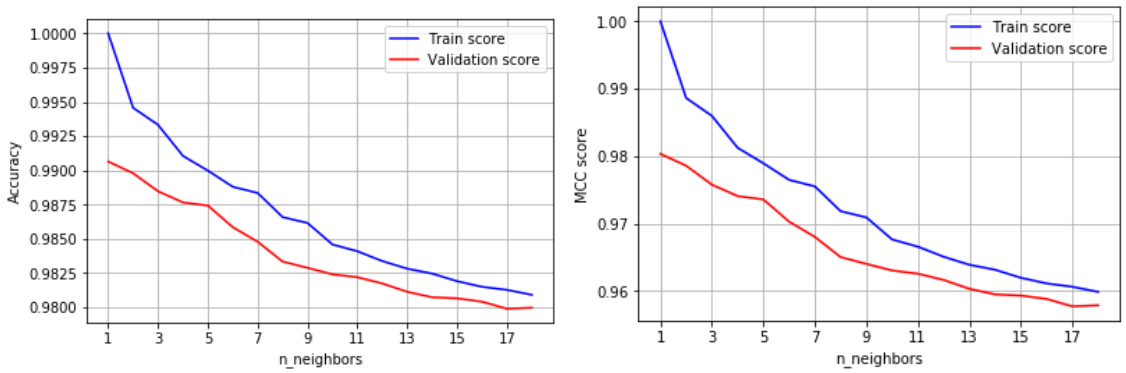


Figura 8: Desempenhos da árvore de decisão variando-se o parâmetro *max_depth*, para acurácia, F1 score e MCC.

4.2 K-Nearest Neighbors

Na classificação a partir dos vizinhos mais próximos, foi variado somente o parâmetro principal, o *n_neighbors*. A forma de ajuste nesse caso é mais delicado, pois como se vê na Figura 9, a princípio a melhor escolha seria *n_neighbors=1*. Na grande maioria dos casos implica em *overfit*. Uma hipótese de explicação para o menor número de vizinhos é o próprio contexto do problema, pois se trata de uma série temporal, e naturalmente, os pontos próximos temporalmente terão grande peso na decisão da classificação de outro ponto. Dessa forma, o melhor valor para esse parâmetro será $n\_neighbors = 2$, na tentativa de minimizar um pouco o *overfit* causado pela escolha de somente um vizinho.



(a) Acurácia (b) MCC

Figura 9: Variação no número de vizinhos para métricas nos conjuntos de treino e validação.

4.3 Rede Neural MLP

Para a rede neural MLP vários parâmetros foram variados afim de encontrar a combinação que obtivesse a maior performance nos resultados. O método de k-fold *cross-validation*, com k=10, também foi utilizado nessa etapa. Além disso para todas as combinações de parâmetros manteve-se um valor fixo de *random state*, para reprodutibilidade do experimento. Na Figura 10 tem-se um resumo das performance das diferentes configurações. O melhor resultado obtido para acurácia foi de $98,32\%$, com camadas intermediárias de arquitetura (20, 20) e tendo a função de tangente hiperbólica como função de ativação.

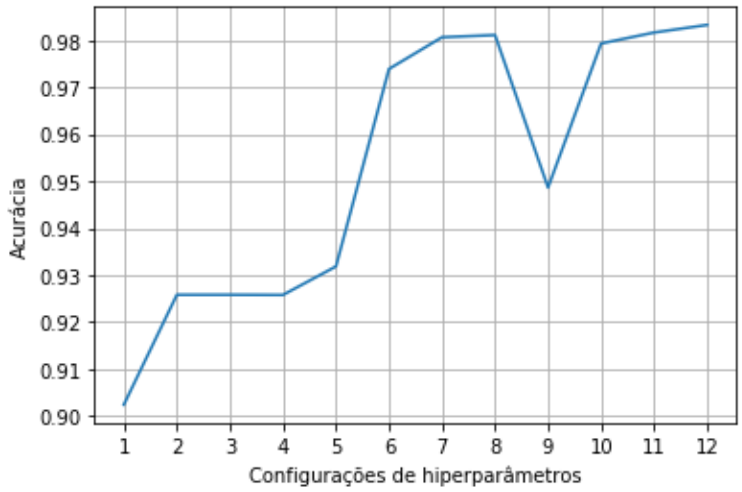
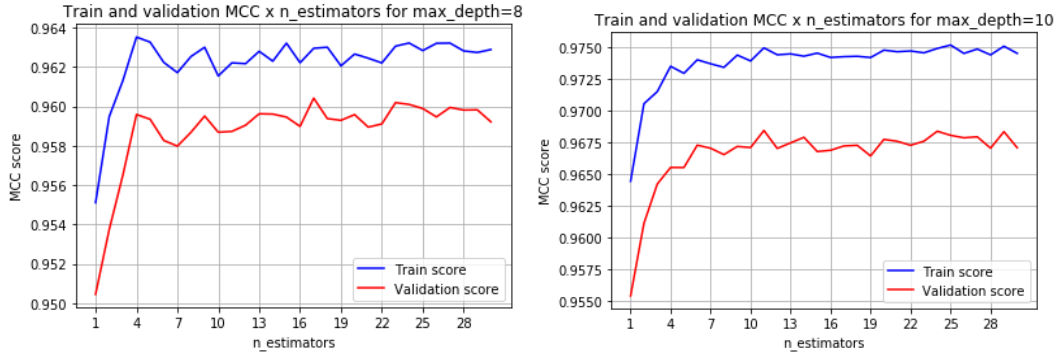


Figura 10: Desempenho de diferentes configurações de hiperparâmetros para o MLP.

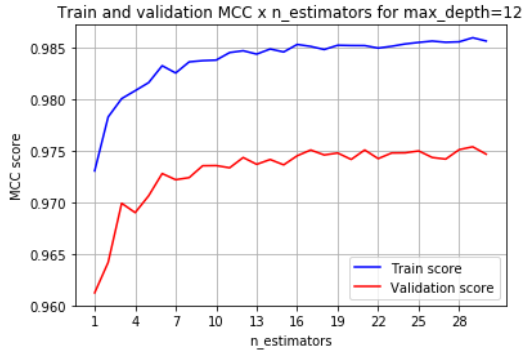
4.4 Random Forest

Em contraste ao caso da árvore de decisão simples, agora temos um outro parâmetro importante a se estimar, o número de estimadores do modelo. Para comparar e comprovar a variação apontada anteriormente com o *max_depth*, o teste será repetido. Os gráficos da Figura 11 apresentam a variação dos parâmetros. Aqui pode-se ver que conforme se aumenta a profundidade da árvore, o distanciamento das performances de treino e validação também aumenta, implicando um *overfit* do modelo. O *trade-off* existente está entre maior valor obtido pelo MCC, e o *overfit* do modelo (especialização somente nos dados de treino). Logo, considerou-se como melhores parâmetros $max\_depth = 12$ e $n\_estimators = 20$. É importante notar que a performance fica constante a partir de valores baixos de $n\_estimators$.

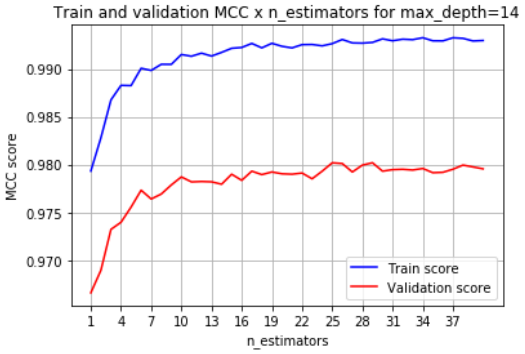


(a) $max\_depth = 8$

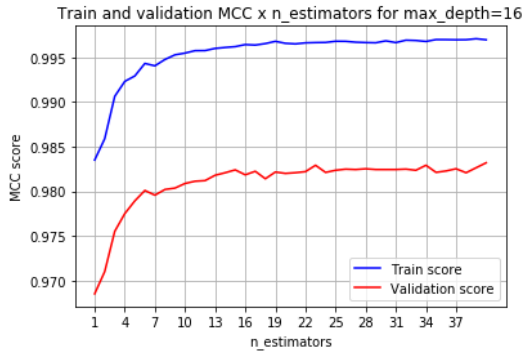
(b) $max\_depth = 10$



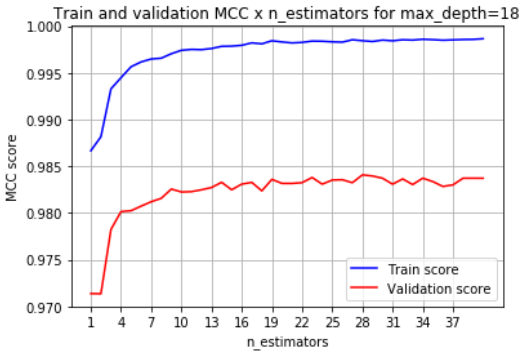
(c) $max\_depth = 12$



(d) $max\_depth = 14$



(e) $max\_depth = 16$



(f) $max\_depth = 18$

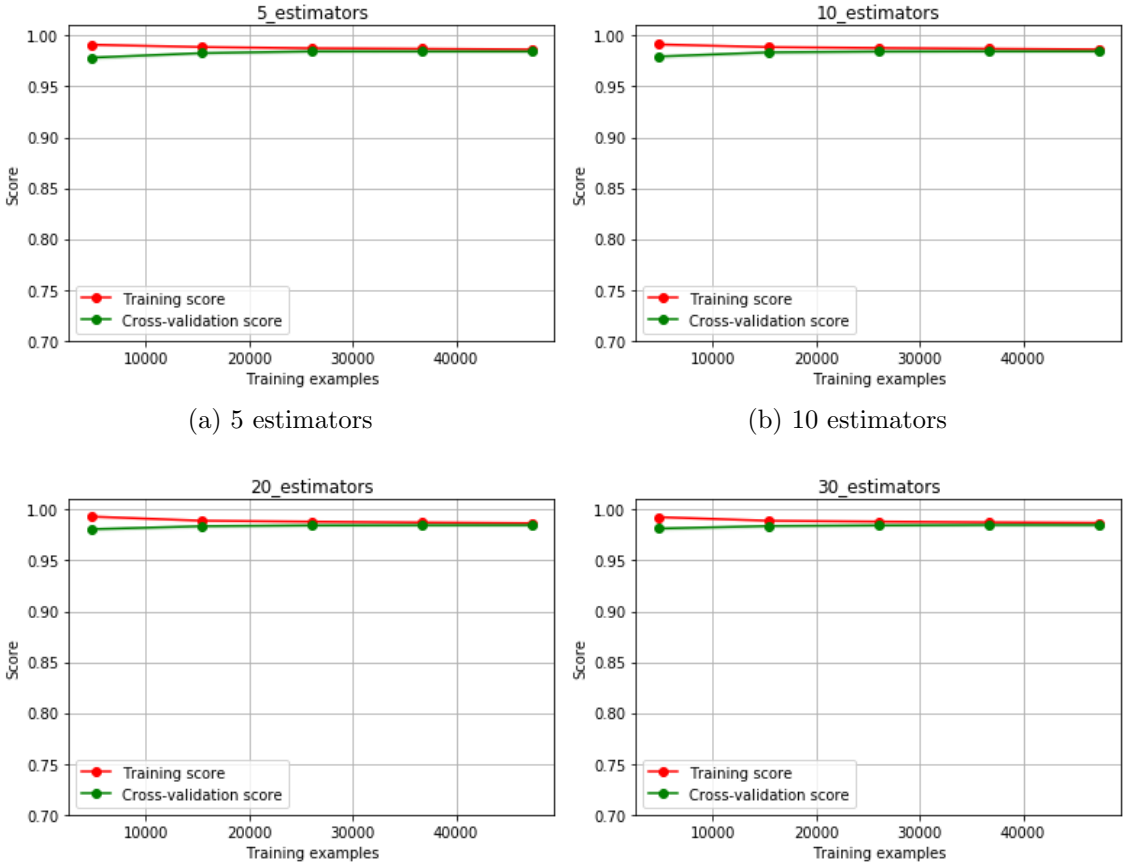
Figura 11: Variação do número de estimadores com max_depth (treino e validação).

4.5 Comitê de MLP

Para o algoritmo de comitê baseado em redes neurais MLP, foram selecionados os parâmetros que obtiveram melhores resultados no classificador MLP simples. O comitê escolhido foi o *bagging* (*Bootstrap Aggregating*). Nesse comitê os classificadores são treinados de forma independente por diferentes conjuntos de treinamento através do método de inicialização, e o conjunto de dados gera um conjunto de modelos utilizando um algoritmo de aprendizagem por meio da combinação por votos para classificação.

Foram testados vários valores para o parâmetro $n\_estimators$, que mos-

traram pouca variação nos resultados, como pode ser observado nas curvas de treinamento apresentadas na Figura 12. Dessa forma, o valor para esse parâmetro considerado melhor foi $n\_estimators = 5$, por ter menor custo.



(a) 5 estimators

(b) 10 estimators

(c) 20 estimators

(d) 30 estimators

Figura 12: Curvas de treinamento para o comitê de MLP.

4.6 Comitê Heterogêneo

O comitê heterogêneo é formado por classificadores de naturezas diferentes. Nesse contexto, foram selecionados os modelos de árvore de decisão, o KNN e o MLP (com suas respectivas melhores configurações). A forma de contribuição de cada classificador é feita com o *Voting Classifier*. Comparando-se o resultado de desempenho dos classificadores individuais e do comitê, tem-se os seguintes valores de MCC:

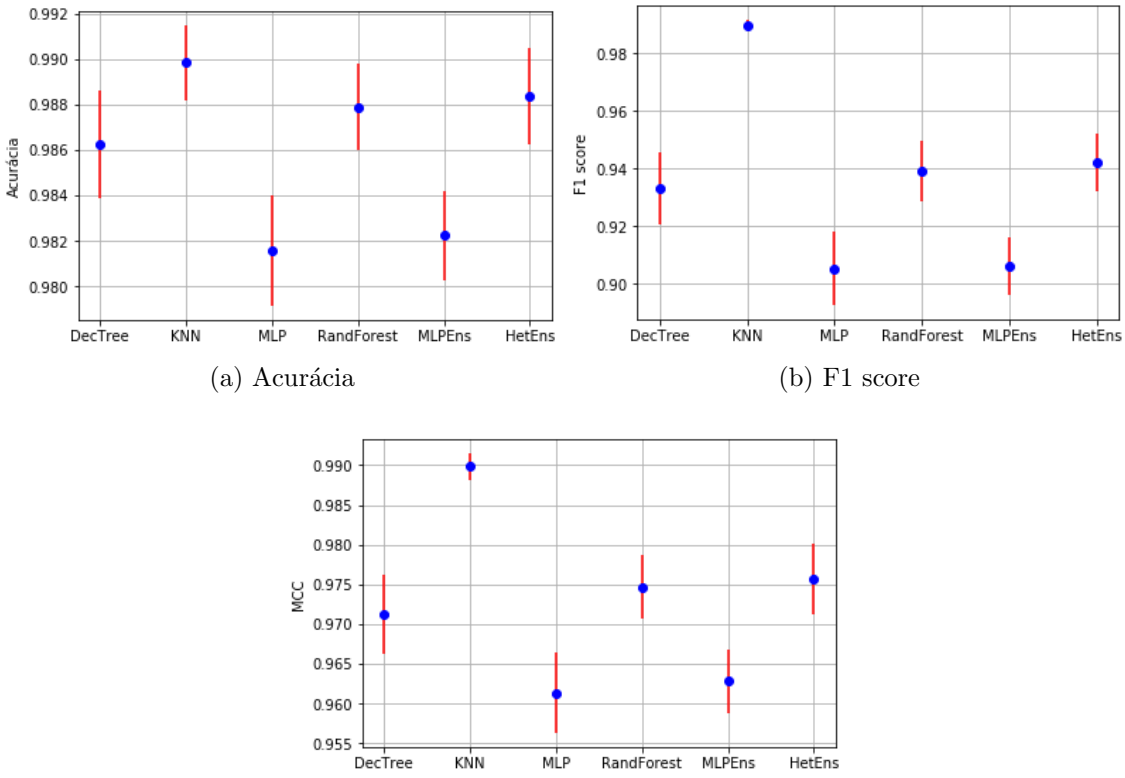
```
Árvore de decisão - 0.9717752
KNN - 0.97851257
MLP - 0.96212
Comitê Heterogêneo: 0.976
```

Dessa forma, mostra-se que o comitê heterogêneo obteve um desempenho intermediário entre os classificadores indivuais.

5 Avaliação

Para a avaliação de desempenho dos algoritmos é interessante que seja explorado mais do que uma métrica. Para algoritmos de classificação, a forma mais geral de se analisar a performance é a partir da matriz de confusão. Nela, está contida todos os exemplos que foram preditos, divididos em falso positivo, falso negativo, verdadeiro positivo e verdadeiro negativo. A partir dessa matriz, são calculadas as métricas acurácia, F1 score e MCC. Considerando as melhores configurações dos modelos, foi feita uma comparação entre os classificadores, utilizando-se de uma validação cruzada k-fold com $k = 20$. Essa escolha para o número de *folds* foi feita para se obter mais amostras para o teste não-paramétrico, que naturalmente tem menos potência, e pela quantidade grande de exemplos disponível, implicando em um número ainda alto para a separação de treino/validação.

Primeiramente, é apresentada a Figura 13, com gráfico de barras de erro, para os valores de média e desvio padrão das performances.



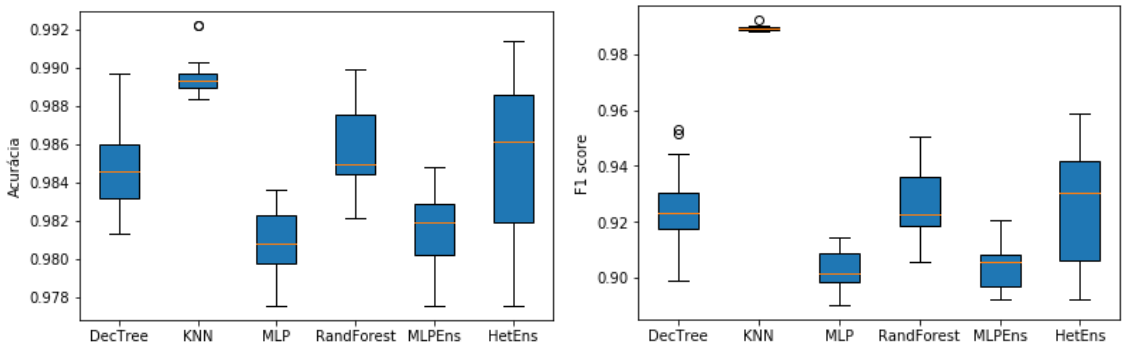
(a) Acurácia

(b) F1 score

(c) MCC

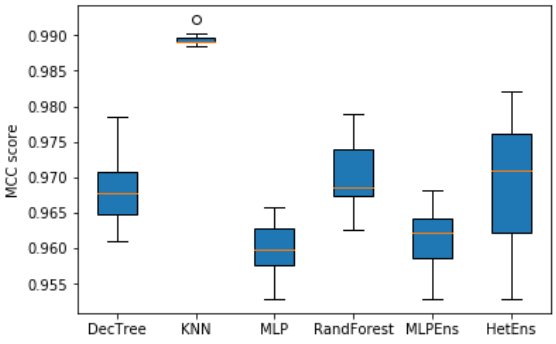
Figura 13: Média e desvio padrão obtidos pela validação cruzada *20-fold* dos modelos.

Outro gráfico que também traz informações sobre as distribuições de valores obtidos, é apresentado na Figura 14.



(a) Acurácia

(b) F1 score



(c) MCC

Figura 14: *Boxplots* obtidos pela validação cruzada *20-fold* dos modelos.

A partir dos dados apresentados nos *boxplots*, já poderia-se dizer quais modelos obtiveram melhor performance. No entanto, dada as variações de performance considerando diferentes dados de treino e outras variações aleatórias, necessita-se de testes estatísticos para se poder afirmar com maior propriedade sobre qual o melhor classificador. De antemão, já vemos que para o F1 e MCC, o desempenho do KNN foi maior do que a de todos os outros casos, e possui pequena variância.

Como se deseja comparar mais de dois modelos de forma pareada, será usado o teste não-paramétrico de Kruskal-Wallis, não assumindo uma forma específica de distribuição dos dados. Para as três métricas abordadas, foi rejeitada a hipótese nula, assumindo-se então a hipótese alternativa de que as distribuições são diferentes:

```
Teste de Kruskal-Wallis para validação cruzada 20-fold

Acurácia: estatística=81.161, p-value=0.00000000000000047964
F1 score: estatística=95.234, p-value=0.00000000000000000053
MCC: estatística=93.056, p-value=0.00000000000000000153

Distribuições diferentes (rejeita H0)
```

Após isso, é feito o teste *post-hoc* de Nemenyi, comparando-se os al-

goritmos de forma pareada (Figura 15). Um destaque merece ser dado para os classificadores KNN e comitê heterogêneo, que obtiveram melhor desempenho, mas nas três diferentes métricas, não foi possível rejeitar a hipótese nula, para um nível de significância de 5%.

1	Árvore de decisão
2	KNN
3	MLP
4	Random Forest
5	Comitê MLP
6	Comitê Heterogêneo

Posthoc Nemenyi for accuracy

	1	2	3	4	5	6
1	-1.000000	4.278506e-02	3.067420e-02	0.817408	7.287382e-02	0.480139
2	0.042785	-1.000000e+00	4.355513e-09	0.609191	3.578990e-08	0.900868
3	0.030674	4.355513e-09	-1.000000e+00	0.000139	9.997870e-01	0.000007
4	0.817408	6.091906e-01	1.387272e-04	-1.000000	5.766781e-04	0.995426
5	0.072874	3.578990e-08	9.997870e-01	0.000577	-1.000000e+00	0.000036
6	0.480139	9.008676e-01	6.798665e-06	0.995426	3.554310e-05	-1.000000

Posthoc Nemenyi for F1 score

	1	2	3	4	5	6
1	-1.000000	1.026202e-03	3.073091e-02	0.970662	3.397480e-02	0.863218
2	0.001026	-1.000000e+00	1.415531e-12	0.025357	1.869587e-12	0.078294
3	0.030731	1.415531e-12	-1.000000e+00	0.001337	1.000000e+00	0.000229
4	0.970662	2.535662e-02	1.337256e-03	-1.000000	1.536817e-03	0.999253
5	0.033975	1.869587e-12	1.000000e+00	0.001537	-1.000000e+00	0.000268
6	0.863218	7.829439e-02	2.292648e-04	0.999253	2.680839e-04	-1.000000

Posthoc Nemenyi for MCC

	1	2	3	4	5	6
1	-1.000000	2.953991e-04	5.677129e-02	0.869864	1.389019e-01	0.733418
2	0.000295	-1.000000e+00	8.081112e-13	0.034400	1.541104e-11	0.075804
3	0.056771	8.081112e-13	-1.000000e+00	0.000653	9.995402e-01	0.000177
4	0.869864	3.439999e-02	6.525745e-04	-1.000000	2.916309e-03	0.999855
5	0.138902	1.541104e-11	9.995402e-01	0.002916	-1.000000e+00	0.000905
6	0.733418	7.580410e-02	1.772567e-04	0.999855	9.053353e-04	-1.000000

Figura 15: Teste *post-hoc* de Nemenyi, para comparação múltipla após o teste de Kruskal-Wallis.

6 Implantação

Apesar de não ser rejeitada a hipótese nula de igualdade das distribuições, o classificador KNN é mais simples e menos custoso computacionalmente, então ele foi escolhido para ser utilizado no conjunto de teste. Como resultado, é apresentada a matriz de confusão na Figura 16, e os valores das três métricas selecionadas para análise:

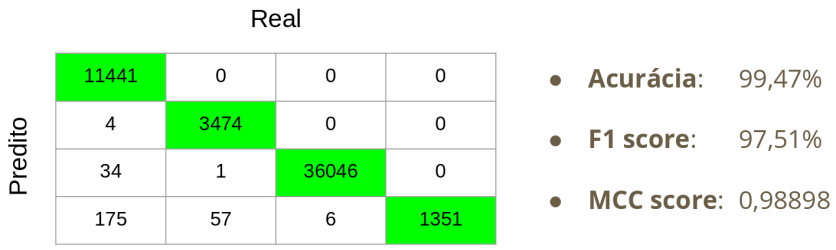


Figura 16: Matriz de confusão para o uso de KNN nos dados de teste.

7 Trabalhos futuros

Apesar de serem obtidos resultados excepcionais, algumas abordagens futuras podem ser consideradas. A seguir são listadas algumas delas, que em algum momento do projeto foram discutidas como implementações que gerariam possíveis melhoras. Outra questão é a geração de uma base de dados que não mescle todos os dados, como feito nesse trabalho.

- Criar uma base de dados separada para cada indivíduo, para poder comparar o desempenho dos classificadores com os artigos que citam essa base de dados;
- Adicionar atributos de séries temporais, como *shifting* e *lagging*;
- Implementação de filtro para o ruído das acelerações;
- Aplicar técnicas para balanceamento das classes, e avaliar tanto o uso de *oversampling*, para aumentar representantes da classe 4, como *undersampling*, para tentar nivelar a quantidade de exemplos.