UNIVERSIDADE FEDERAL DO CEARÁ

Reconhecimento de Padrões

João Rafael Barbosa de Araujo
6 de maio de 2019

Esse relatório contém as respostas para o segundo trabalho da disciplina de Reconhecimento de padrões. Além de um curto resumo das diferentes técnicas de validação e classificação utilizados no trabalho, assim como os resultados das diferentes técnicas aplicadas ao banco de dados *iris_log* e *dermatology* de acordo com o que foi solicitado nas questões.

1 PRIMEIRA QUESTÃO

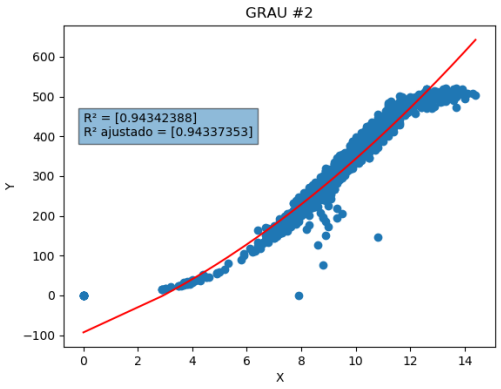
Usando o conjunto de dados do aerogerador (variável de entrada: velocidade do vento – m/s, variável de saída: potência gerada – kWatts), determine o modelo de regressão polinomial (graus 2, 3, 4, 5 e 6) com parâmetros estimados pelo método dos mínimos quadrados. Avaliar a qualidade de cada modelo por meio do coeficiente de determinação (R^2).

Regressão polinomial é uma regressão feita a partir de combinações não lineares das entradas. Essa regressão é utilizada para tentar representar dados que não são linearmente comportados, em que é estendido o conceito de regressão linear a partir da criação de novas variáveis polinomiais a partir de combinações das entradas originais.

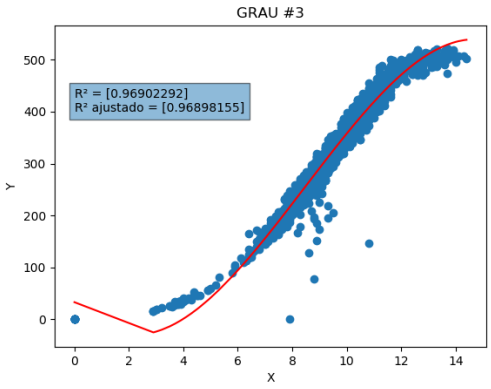
Na Figura 1.1 podemos observar que para esse conjunto de dados, aumentar o grau do polinômio resulta em uma melhor aproximação do comportamento dos pontos. Para avaliar a qualidade dessa regressão é utilizado o coeficiente de determinação R^2. Porém R^2 não reflete bem a qualidade da regressão, por essa razão alguns autores preferem utilizar R^2 ajustado onde há uma penalização pelo acréscimo de novas variáveis, podendo assim definir um balanço entre qualidade do regerssor e tamanho do vetor de variáveis de regressão.

Podemos observar que para os graus 2, 3 e 4 o valor de R^2 subiu a cada novo grau, já no quinto grau vemos que o valor de R^2 ajustado diminuiu. Quando há a diminuição do valor de R^2 ajustado é sinal que o grau anterior é o ideal para esse conjunto de dados e que adicionar mais variáveis não vai melhorar significativamente o regressor.

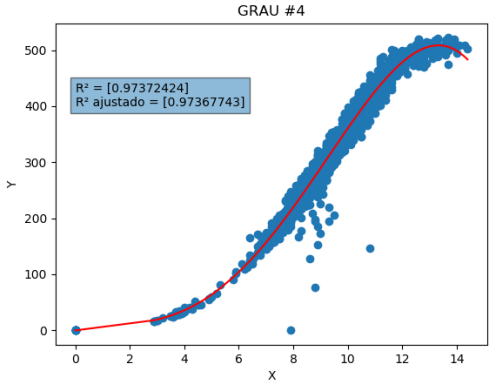
O código utilizado para gerar os resultados pode ser encontrado em *rp_regressao_q1.py*.



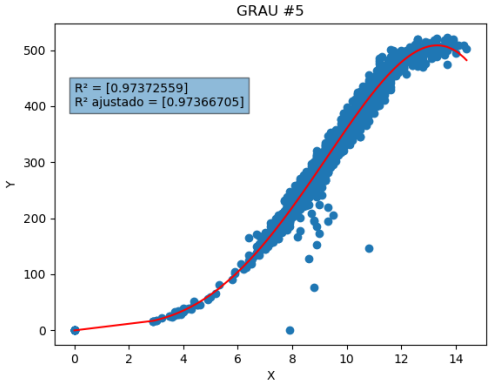
(a) Regressão polinomial de grau 2



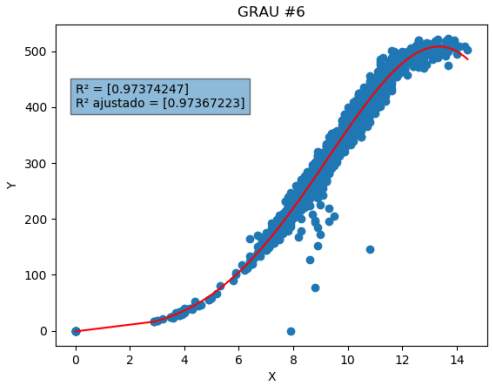
(b) Regressão polinomial de grau 3



(c) Regressão polinomial de grau 4



(d) Regressão polinomial de grau 5



(e) Regressão polinomial de grau 6

Figura 1.1: Regressão polinomial para diferentes graus e respectivos R²

2 SEGUNDA QUESTÃO

Dada a base de dados abaixo, na qual a primeira e segunda colunas são as variáveis regressoras (x1 e x2) e a terceira coluna é a variável dependente (y), determine o modelo de regressão múltipla (plano) com parâmetros estimados pelo método dos mínimos quadrados. Avalie a qualidade do modelo por meio do coeficiente de determinação (R^2).

D =[122 139 0.115;
114 126 0.120;
086 090 0.105;
134 144 0.090;
146 163 0.100;
107 136 0.120;
068 061 0.105;
117 062 0.080;
071 041 0.100;
098 120 0.115];

Similar ao que foi feito na questão 1, aqui também iremos calcular diferentes valores de R^2 ajustado para esse conjunto de dados. Uma diferença para esse bando de dados é que agora temos 2 valores de X para cada amostra.

```
GRAU = 1
R² = [0.72388235]
R² ajustado = [-0.1044706]
---------------------------
GRAU = 2
R² = [0.83532045]
R² ajustado = [1.6587182]
---------------------------
GRAU = 3
R² = [0.83928699]
R² ajustado = [1.21428401]
---------------------------
GRAU = 4
R² = [0.77426695]
R² ajustado = [1.18058644]
---------------------------
GRAU = 5
R² = [0.96537463]
R² ajustado = [1.01978593]
---------------------------
GRAU = 6
R² = [1.]
R² ajustado = [1.]
```

Para esses dados o valor de R^2 aumenta até chegar em 1 no sexto grau. Já o comportamento de R^2 ajustado aumenta inicialmente, e então passa a diminuir a partir do segundo grau. Assim, de acordo com R^2 ajustado é interessante utilizar até o segundo grau apenas.

O código utilizado para gerar os resultados pode ser encontrado em *rp_regressao_q2.py*.

3 TERCEIRA QUESTÃO

Classificar o conjunto de dados Dermatology disponível em http://archive.ics.uci.edu/ml/datasets.php, conforme as instruções a seguir.

- **Classificador: ELM, Perceptron, MLP (implementar os dois primeiros classificadores);**
- **Estratégia de validação cruzada: leave-one-out e 5-fold;**
- **Testar com e sem o zscore;**
- **Montar a matriz de confusão para cada classificação usando os dados de testes.**

3.1 PERCEPTRON

Para a implementação do algoritmo de perceptron foi utilizado o livro *Hands-On Machine Learning with Scikit-Learn & TensorFlow*, onde foi mostrado que existem duas funções de ativação do neuronio que são comumente utilizadas na literatura. Para essa implementação a função utilizada foi **heaviside()** que retorna 1 para qualquer elemento maior que zero ou retorna zero caso contrário.

Como a rede perceptron tem apenas um neurônio, foi utilizada a técnica *one versus all*, onde a rede foi avaliada para uma classe por vez, ou seja, é testada a capacidade da rede de diferenciar uma classe em relação a todas as outras.

3.1.1 LEAVE-ONE-OUT

Como foi falado na introdução dessa seção, os algoritmos de perceptron serão avaliados utilizando *one vs all*. Existem alguns parâmetros de parada que podem ser ajustados dentro do algoritmo da *Perceptron*, dentre eles o numero de atualizações dos peses ou um ϵ que para a simulação quando o crescimento da aprendizagem se torna muito pequeno.

Para essa simulação foi utilizado o numero de atualizações como critério de parada, esse valor é chamado de *epochs* (do inglês, época).

Abaixo temos a classificação de cada classe de acordo com um certo numero de épocas (sem normalização).

```
For 4 epochs:
for class:1 Result:98.04 %
for class:2 Result:92.46 %
for class:3 Result:99.72 %
for class:4 Result:94.69 %
for class:5 Result:98.88 %
for class:6 Result:100.00 %
---------------------------
For 5 epochs:
for class:1 Result:97.77 %
for class:2 Result:90.78 %
for class:3 Result:99.44 %
for class:4 Result:94.41 %
for class:5 Result:99.72 %
for class:6 Result:100.00 %
---------------------------
For 6 epochs:
for class:1 Result:98.32 %
for class:2 Result:93.85 %
for class:3 Result:100.00 %
```

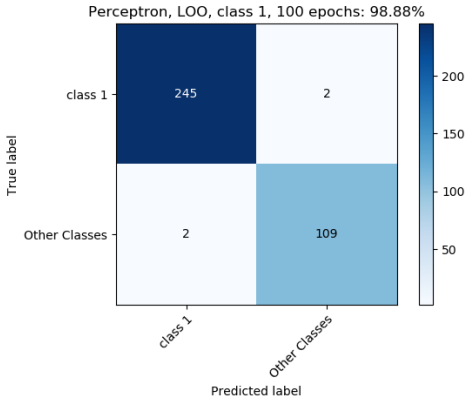
```
for class:4 Result:94.13 %
for class:5 Result:99.72 %
for class:6 Result:100.00 %
----------------------------
For 7 epochs:
for class:1 Result:98.60 %
for class:2 Result:93.58 %
for class:3 Result:100.00 %
for class:4 Result:94.97 %
for class:5 Result:100.00 %
for class:6 Result:100.00 %
----------------------------
For 8 epochs:
for class:1 Result:98.88 %
for class:2 Result:94.41 %
for class:3 Result:100.00 %
for class:4 Result:95.25 %
for class:5 Result:99.44 %
for class:6 Result:100.00 %
```

Abaixo temos a classificação de cada classe de acordo com um certo numero de épocas (com normalização).

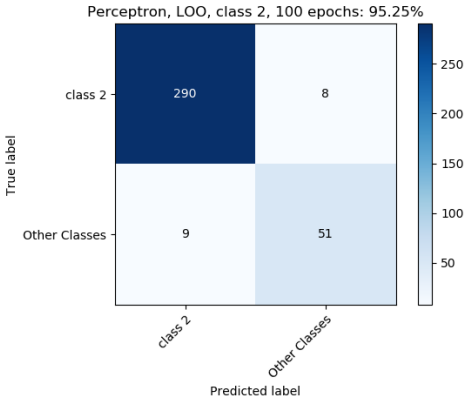
```
For 1 epochs:
for class:1 Result:99.72 %
for class:2 Result:99.72 %
for class:3 Result:100.00 %
for class:4 Result:99.44 %
for class:5 Result:99.44 %
for class:6 Result:99.44 %
----------------------------
For 2 epochs:
for class:1 Result:100.00 %
for class:2 Result:99.72 %
for class:3 Result:100.00 %
for class:4 Result:100.00 %
for class:5 Result:100.00 %
for class:6 Result:100.00 %
----------------------------
For 3 epochs:
for class:1 Result:100.00 %
for class:2 Result:100.00 %
for class:3 Result:100.00 %
for class:4 Result:99.72 %
for class:5 Result:99.72 %
for class:6 Result:99.72 %
----------------------------
For 4 epochs:
for class:1 Result:100.00 %
for class:2 Result:100.00 %
for class:3 Result:100.00 %
for class:4 Result:100.00 %
for class:5 Result:100.00 %
for class:6 Result:100.00 %
----------------------------
For 5 epochs:
for class:1 Result:100.00 %
for class:2 Result:100.00 %
for class:3 Result:100.00 %
for class:4 Result:100.00 %
for class:5 Result:100.00 %
for class:6 Result:100.00 %
----------------------------
```

Outra forma de observar o desempenho da rede é através da matriz de confusão. A matriz de confusão é uma matriz quadrada de lado igual ao número de classes, e o valor de cada célula é a quantidade de vezes que a classe real foi igual a classe estimada.

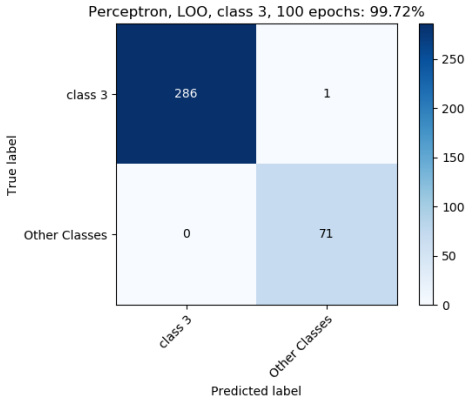
Na figura 3.1 temos as matrizes de confusão para 100 épocas para cada classe.



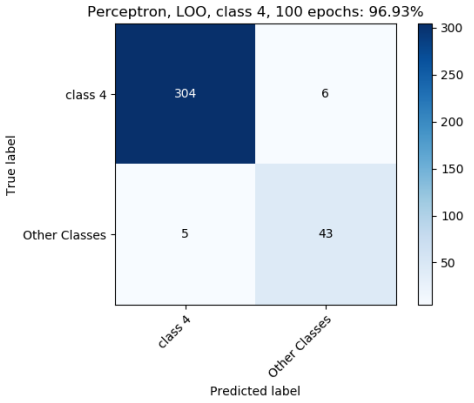
(a) Matriz de confusão para a classe 1



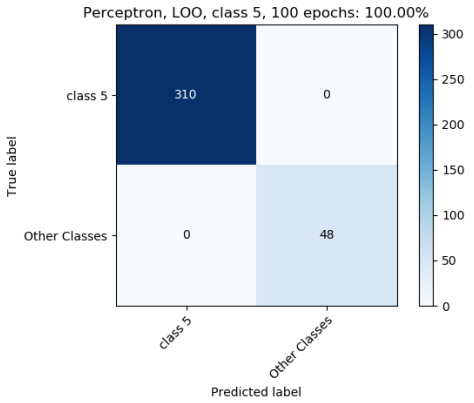
(b) Matriz de confusão para a classe 2



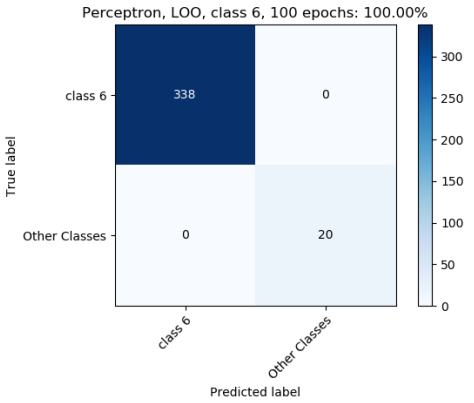
(c) Matriz de confusão para a classe 3



(d) Matriz de confusão para a classe 4



(e) Matriz de confusão para a classe 5



(f) Matriz de confusão para a classe 6

Figura 3.1: Matrizes de confusão por classe utilizando *Leave-one-out*

.

3.1.2 '5-FOLD' (K-FOLD)

Assim como na subseção anterior, por estarmos utilizando uma rede perceptron, a classificação é feita individualmente para cada classe.

Abaixo estão os resultados variando a quantidade de épocas para cada classe (sem normalização):

```
For 1 epochs:
for class:1 Result:54.19 %
for class:2 Result:69.83 %
for class:3 Result:63.97 %
for class:4 Result:72.07 %
for class:5 Result:72.07 %
for class:6 Result:85.47 %
----------------------------
For 11 epochs:
for class:1 Result:84.36 %
for class:2 Result:77.93 %
for class:3 Result:97.49 %
for class:4 Result:66.48 %
for class:5 Result:92.74 %
for class:6 Result:97.21 %
----------------------------
For 21 epochs:
for class:1 Result:86.87 %
for class:2 Result:79.33 %
for class:3 Result:100.00 %
for class:4 Result:83.24 %
for class:5 Result:95.81 %
for class:6 Result:98.04 %
----------------------------
For 31 epochs:
for class:1 Result:95.81 %
for class:2 Result:74.30 %
for class:3 Result:100.00 %
for class:4 Result:82.12 %
for class:5 Result:98.60 %
for class:6 Result:98.60 %
----------------------------
For 41 epochs:
for class:1 Result:96.93 %
for class:2 Result:86.03 %
for class:3 Result:100.00 %
for class:4 Result:89.94 %
for class:5 Result:97.21 %
for class:6 Result:98.88 %
----------------------------
For 51 epochs:
for class:1 Result:96.65 %
for class:2 Result:85.75 %
for class:3 Result:100.00 %
for class:4 Result:81.28 %
for class:5 Result:98.60 %
for class:6 Result:98.88 %
```

Abaixo estão os resultados variando a quantidade de épocas para cada classe (com normalização):

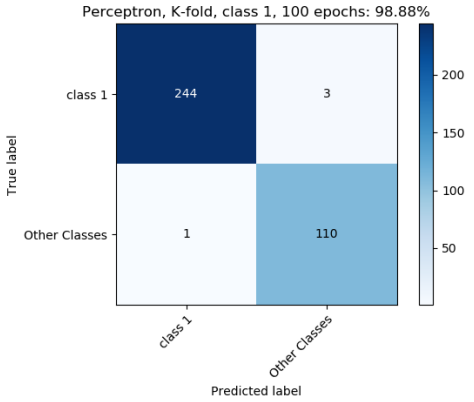
```
For 1 epochs:
for class:1 Result:98.60 %
for class:2 Result:89.94 %
for class:3 Result:99.72 %
for class:4 Result:83.24 %
for class:5 Result:90.78 %
for class:6 Result:89.39 %
For 11 epochs:
for class:1 Result:99.16 %
for class:2 Result:98.88 %
for class:3 Result:100.00 %
for class:4 Result:99.72 %
for class:5 Result:100.00 %
for class:6 Result:99.72 %
----------------------------
For 21 epochs:
```

```
for class:1 Result:98.88 %
for class:2 Result:98.88 %
for class:3 Result:100.00 %
for class:4 Result:99.16 %
for class:5 Result:100.00 %
for class:6 Result:99.72 %
----------------------------
For 31 epochs:
for class:1 Result:98.88 %
for class:2 Result:100.00 %
for class:3 Result:100.00 %
for class:4 Result:99.44 %
for class:5 Result:100.00 %
for class:6 Result:99.72 %
----------------------------
For 41 epochs:
```

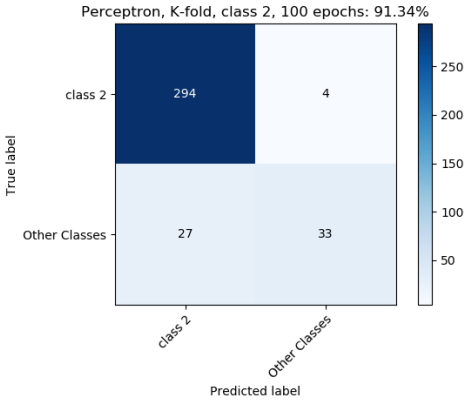
```
for class:1 Result:98.88 %
for class:2 Result:100.00 %
for class:3 Result:100.00 %
for class:4 Result:99.72 %
for class:5 Result:100.00 %
for class:6 Result:99.72 %
----------------------------
For 51 epochs:
for class:1 Result:98.88 %
for class:2 Result:100.00 %
for class:3 Result:100.00 %
for class:4 Result:99.72 %
for class:5 Result:100.00 %
for class:6 Result:99.72 %
----------------------------
```

Outra forma de observar o desempenho da rede é através da matriz de confusão. A matriz de confusão é uma matriz quadrada de lado igual ao número de classes, e o valor de cada célula é a quantidade de vezes que a classe real foi igual a classe estimada.

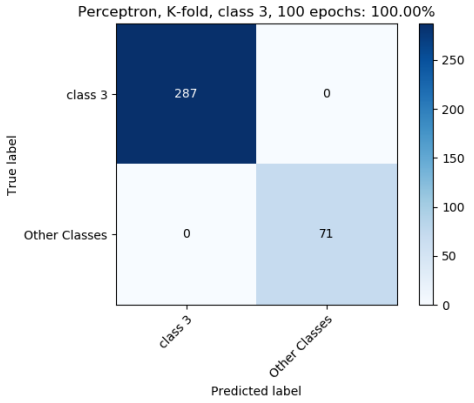
Na figura 3.1 temos as matrizes de confusão para 100 épocas para cada classe.



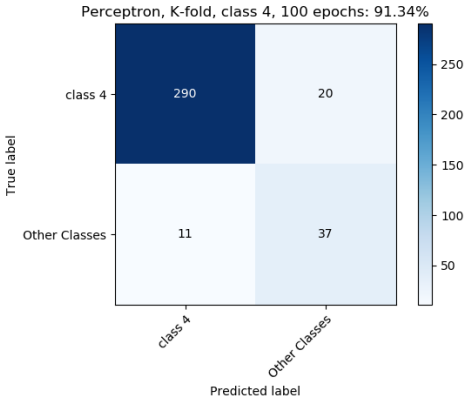
(a) Matriz de confusão para a classe 1



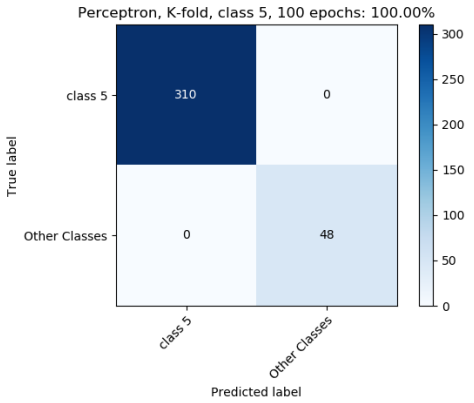
(b) Matriz de confusão para a classe 2



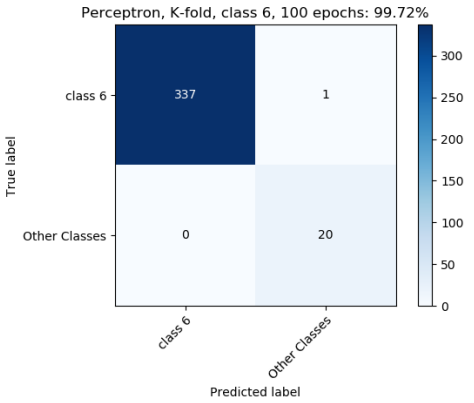
(c) Matriz de confusão para a classe 3



(d) Matriz de confusão para a classe 4



(e) Matriz de confusão para a classe 5



(f) Matriz de confusão para a classe 6

Figura 3.2: Matrizes de confusão por classe utilizando *k-fold*

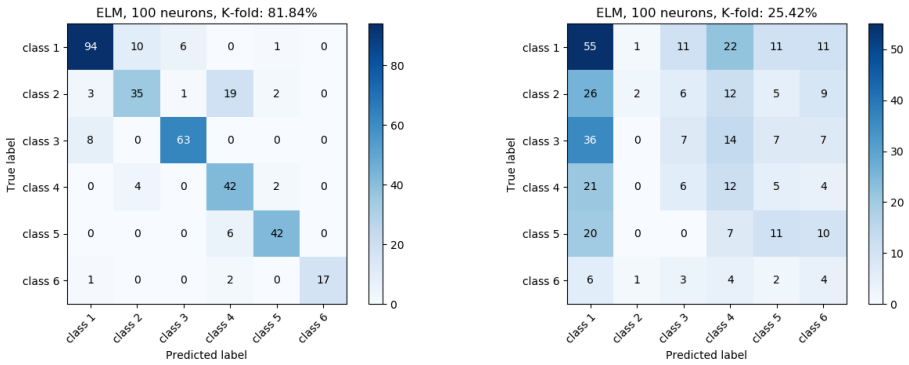
.

3.2 EXTREME LEARNING MACHINE

Nessa subseção iremos mostrar os resultados utilizando ELM. Para o desenvolvimento do código do ELM foram utilizados apenas os slides dados em sala de aula como referência. Sua implementação é bastante direta e a fase de treinamento não funciona de forma iterativa. Na rede ELM há apenas uma camada oculta, que é inicializada com neuronios aleatorios.

3.2.1 K-FOLD

Podemos observar na Figura 3.3 que para a ELM é importante que os dados estejam normalizados. O desempenho da rede caiu em mais de 50% para os dados não normalizados, tendo seu melhor resultado acertando 81.84% e o pior desempenho acertando apenas 25.42%.



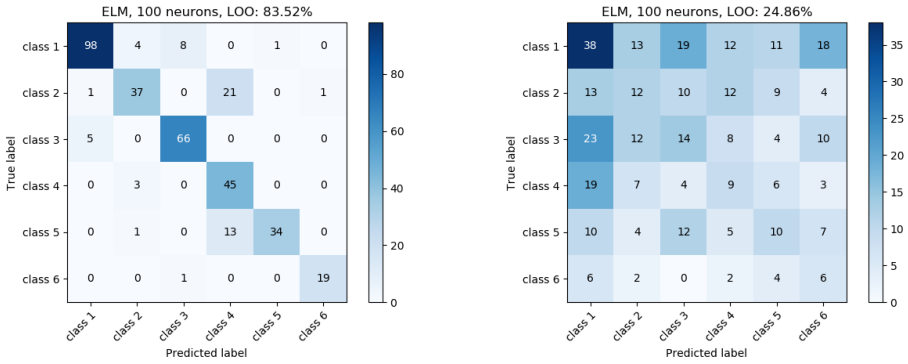
(a) Matriz de confusão com entrada normalizada

(b) Matriz de confusão com entrada não normalizada

Figura 3.3: Matrizes de confusão utilizando *k-fold*

3.2.2 LEAVE-ONE-OUT

Utilizando leave-one-out também é possível observar (Figura 3.4) que a normalização dos dados gera resultados bem mais significativos em relação aos dados não normalizados. Aqui o melhor resultado foi 83.52%, marginalmente melhor que utilizando k-fold como validação. O pior resultado ficou com uma acurácia de 24.86%, mostrando desempenho pior que a validação cruzada anterior.



(a) Matriz de confusão com entrada normalizada

(b) Matriz de confusão com entrada não normalizada

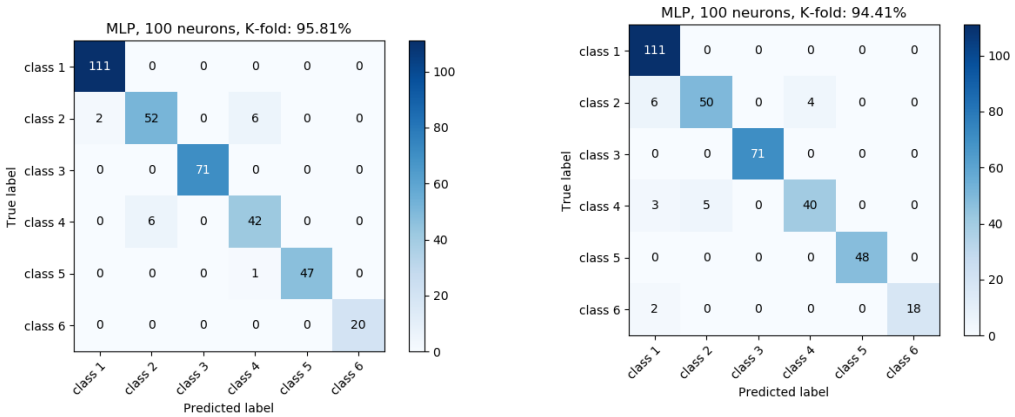
Figura 3.4: Matrizes de confusão utilizando *k-fold*

3.3 MULTI-LAYER PERCEPTRON

Nessa para os item utilizando MLP não foi necessário implementar a rede, por essa razão utilizei a biblioteca *scikit-learn - Machine Learning in Python* para aplicar MLP no banco de dados.

3.3.1 K-FOLD

A rede MLP obteve os melhores resultados em relação a normalização dos dados. É possível observar que essa rede é mais resistente a dados não normalizados. Apesar de uma pequena perda de desempenho em dados não normalizados, foi possível uma acurácia de 95.81% no melhor caso. Já nos dados não normalizados a acurácia caiu para 94.41%.



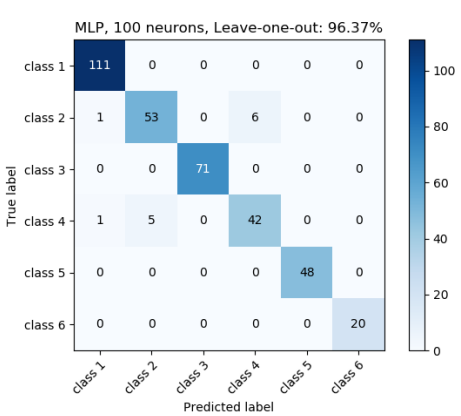
(a) Matriz de confusão com entrada normalizada

(b) Matriz de confusão com entrada não normalizada

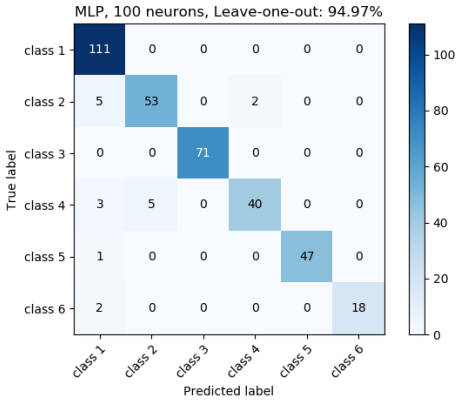
Figura 3.5: Matrizes de confusão utilizando *k-fold*

3.3.2 LEAVE-ONE-OUT

Similar ao método de validação cruzada acima, a MLP também obteve bons resultados em relação a normalização dos dados. Seu desempenho ficou em 96.37% para os dados normalizados e 94.97% para o caso não normalizado.



(a) Matriz de confusão com entrada normalizada



(b) Matriz de confusão com entrada não normalizada

Figura 3.6: Matrizes de confusão utilizando *k-fold*

4 CÓDIGO

Nessa seção será apresentada informações sobre como foram feitos os algoritmos e também os resultados desses algoritmos aplicados nas bases de dados para cada questão.

4.1 PROGRAMAÇÃO

A linguagem de programação utilizada para resolver o problema foi *Python 3.7*. Foram utilizadas as bibliotecas *sklearn* e *numpy* para auxiliar na aplicação das técnicas discutidas na seções 1 2 e 3, e também no tratamento de dados.

A base de dados “aerogerador.dat” contém 2250 amostras mapeando uma velocidade de vento para uma produção de energia.. Cada amostra da base contém apenas uma entrada e uma saída.

A base de dados "Dermatology.dat"contém 358 amostras divididas em 6 classes. Cada amostra contém 34 *features*.

5 CONCLUSÃO

Nesse relatório foram utilizados diferentes métodos de classificação, que foram testados utilizando 2 métodos de validação cruzada. As técnicas de classificação foram: MLP, ELM e

perceptron. Existe um compromisso entre as diferentes técnicas, onde para bancos de dados diferentes pode haver técnicas de classificação mais adequadas.

A normalização dos dados se mostrou bastante eficiente para algumas questões, levanta a taxa de acerto a valores mais de 50% melhores. Porém na MLP sua influencia no resultado se mostrou bem menos efetiva..

Todo o código desenvolvido para esse trabalho pode ser encontrado em `https://github.com/faellacurcio/rp`.

6 APÊNDICE

6.1 VALIDAÇÃO CRUZADA

Validação cruzada é uma de várias técnicas que é utilizada para estimar a qualidade de generalização de um algoritmo a partir da separação de um conjunto de dados. Seu principal objetivo é quantificar a eficiência de um modelo para a chegada de novos dados além de permitir a detecção de problemas como *overfitting* (uma especialização excessiva no conjunto de dados perdendo assim a capacidade de generalização do modelo) . Em um modelo preditivo é típico haver dois grupos de dados: Dados de treinamento, dados de teste. Há diversas técnicas comumente utilizadas para realizar a validação cruzada.

Validação cruzada é composta pela decomposição de um conjunto de dados em subconjuntos de treinamento e teste, podendo também incluir subconjunto de validação. O primeiro subconjunto, de treinamento, será utilizado para alimentar o modelo. Já o subconjunto, de teste, será utilizado para avaliar a porcentagem de acerto do modelo comparado a classificação do modelo com a classificação real. O ultimo conjunto, de validação, é utilizado em alguns classificadores como referencia para detectar *overfitting* como é feito em redes neurais. Uma regra de ouro da validação cruzada diz que os conjuntos de treinamento, teste e validação não devem possuir elementos em comum! As sub-sessões abaixo exemplificam os métodos de validação cruzada utilizadas no trabalho de Reconhecimento de padrões.

6.2 *Leave-one-out*

Esse método de é um dos mais simples de ser implementado. Ele consistem em utilizar o conjunto completo menos uma amostra para o subconjunto de treinamento, e apenas essa amostra que não foi incluída no grupo de treinamento para ser o subconjunto teste. Para um conjunto de N pontos, são utilizados N-1 amostas no subconjuntos de treinamento e 1 amostra no subconjunto de teste como é mostrado na Figura 6.1. A eficiência do modelo é dada em porcentagem pela média de acertos.

6.3 *k-fold*

O *k-fold* pode ser visto como uma extensão do *leave-one-out* porém ao invés de utilizar uma única amostra para o conjunto de teste, os dados serão divididos em K conjuntos de aproximadamente mesmo tamanho. Similarmente ao que foi mostrado na sub-seção anterior, esse método é utilizado para estimar o quão eficaz o modelo será ao encontrar dados novos,

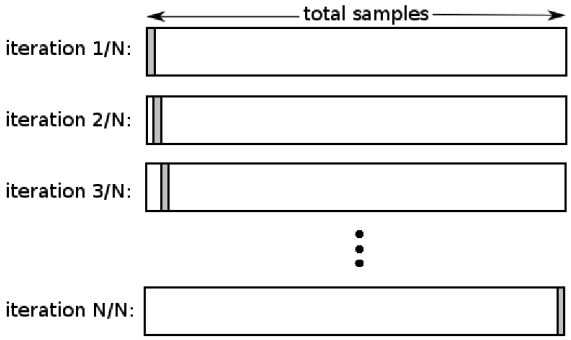


Figura 6.1: Validação cruzada *leave-one-out*

ou seja, quantificar a eficiência de uma técnica de classificação. A figura 6.2 mostra a divisão dos dados, onde cada bloco (*train/test*) contém **k** conjuntos de aproximadamente mesmo tamanho.

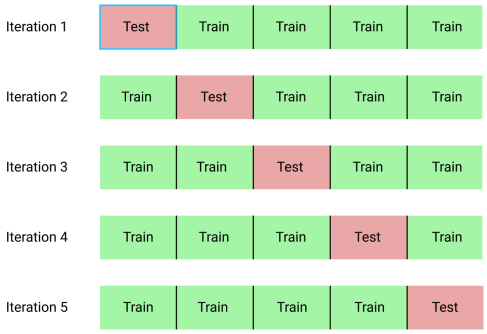


Figura 6.2: Validação cruzada *k-fold*