Detetor de Stress

Raul Barbosa[1],
[1] Departamento Engenharia Informática, Universidade de Coimbra
raulbarbosa1996@gmail.com

Abstract

No âmbito da cadeira de Inteligência Artificial, uma unidade curricular do Mestrado de Engenharia Informática foi realizado um trabalho que, através da criação de uma aplicação *android*, permite detetar/prever se um utilizador tem stress. Além desta aplicação foi desenvolvido um modelo aplicado a um dataset com o intuito de prever uma *feature*, Emotional Stability.

1 Introdução

Este trabalho é composto por duas partes, sendo a primeira o desenvolvimento de uma aplicação que contém um sensor que capta dados sobre o ambiente que rodeia o utilizador. Este dados captados são de certa forma relacionados com o stress e irá ser explicado como foi escolhido o tipo sensor responsável para a deteção de stress e como foi implementado. A segunda parte que irá ser apresentada é o desenvolvimento de um modelo e irá ser explicado que *dataset* foi estudado e com que finalidade foi desenvolvido. Além de falar destes dois modelos é apresentada um pouco da pesquisa feita acerca de aplicações capazes de detetar stress e os modelos O-MaSE.

2 Estado de Arte

2.1 Stress Detection Using Smart Phone Data

Este trabalho utiliza uma tecnologia que reconhece níveis de stress através dos dados dos dispositivos sendo estes: padrões de níveis de sono, vida social e atividade física. A ideia do sistema *StayActive* é a combinação de um modelo matemático e uma abordagem de *Machine Learning* que funcionam independentemente para calcular o nível de stress dos utilizadores. Esta aplicação é sincronizada com a abordagem *Machine Learning* em execução no servidor do sistema *StayActive*. Os resultados destes dois modelos são combinados e resultam num nível de stress final para os utilizadores. Neste estudo, adota-se uma abordagem pragmática e é construído um sistema inicial de deteção de stress que pode ser ampliado e refinado. A arquitetura do sistema pode ser explicada através de 3 pontos: Provedores- responsáveis por a coleção dos dados, tipo de atividade física, chamadas/SMS, luz ambiente, temperatura, localização, numero de toques no ecrã, contador de passos e aplicações executadas; Servidor- responsável por receber os dados dos dispositivos móveis e armazena-los numa base de dados. Aqui são agregados todos os dados e são processados. A abordagem à *Machine Learning* é implementada neste módulo, utilizando os recursos fornecidos pelos provedores que por fim vai calcular um nível de stress; Deteção: Este último modulo contem analisadores para cada provedor de dados, que extraem informações e padrões úteis dos dados brutos para gerar uma pontuação parcial de relaxamento. Por fim, este modulo agrega os resultados e calcula um nível de stress final. Além de coletar tantos dados quanto possível do dispositivo móvel, o sistema solicita que o utilizador preencha um questionário com a sua auto perceção. Para este questionário foi usado o *Circumplex Model of Affect* escrito por James A. Russel. Este modelo consiste em duas dimensões: o prazer-desgosto e a dimensão do despertar-dormir. Alem destas duas dimensões foi acrescentada uma terceira, relaxada. A razão de ter sido escolhido este modelo foi porque representava uma gama mais ampla de estados de humor e era fácil de ser preenchido pelos utilizadores sem que fosse invasivo. Para detetar quantas horas de sono o utilizador teve é calculado os toques no ecrã do dispositivo. Entre as 18:00 e as 10 da manha calcula-se o maior intervalo de tempo onde o utilizador não tocou no ecrã e esse intervalo resulta na duração do sono do utilizador. Assume-se que o número de horas de sono normais é de 8 horas. Por isso é estabelecido um limite inferior e superior de horas de sono sendo o inferior 7 horas e o superior 9 horas. Qualquer valor fora do intervalo é atribuída uma penalização ao utilizador com um fator de peso por hora. No caso da interação social do utilizador, é relevante para esta aplicação o número de toques no ecrã, o número de chamadas e SMS. O resultado por dia é multiplicado pelo fator de ponderação correspondente e é acumulado na pontuação total de relaxamento. A ideia para este calculo consiste em atribuir um fator de peso a cada uma das sub-dimensões que consideram pessoalmente mais importantes para comunicar com as pessoas. Para a dimensão mais importante atribui-se um peso de w1=0,4 e para as restantes de 0,3 (w2=w3=0,3) de modo a que w1+w2+w3=1. Para avaliar a atividade física o sistema faz a distinção entre o tipo de atividade. É implementado um contador de passos que quando atinge a meta dos 10,000 passos por dia é lhe atribuído o valor máximo de bem-estar. Se alguém atingir menos do que esse número é diminuído a pontuação do relaxamento, com um fator de peso de 1,000

passos. Após a receção dos dados por mês e com base no *Circumplex Model of Affect* extrai-se o padrão entre a atividade física ideal de cada utilizador e os passos diários. Esse valor extraído é considerado o valor máximo de bem-estar para esse utilizador e todos os valores são comparados com esse. O cálculo final é feito numa escala de 0-10 onde 0 é muito stressado e 10 relaxado. A ideia do procedimento de pontuação é igual à mesma com a atribuição de pontuação das sub-dimensões da interação social. É calculado um resultado por dimensão. Para cada uma das três dimensões normalizou-se os resultados na escala de [0-10] e depois multiplicou-se cada um deles com o respetivo fator. Portanto, para calcular o resultado numa escala comum, respeitou-se o seguinte procedimento:

1. Primeiro calcula-se os valores padronizados dos itens. Esse valor também é chamado de desvio normal e representa a distância de um ponto de dados da média, dividida pelo desvio padrão da distribuição. std vl = (unstd vl-mean) / x (1). Onde std vl é o valor padronizado e x é o desvio padrão.
2. Em segundo lugar, utiliza-se pesos fatoriais para calcular a pontuação não padronizada. unstd sc = w1 * dm1 + w2 * dm2 + w3 * dm3 (2) onde unstd sc é a pontuação não padronizada, wi é a pontuação do peso do item i e dmi é o valor padronizado do item i.
3. Encontramos o mínimo e o máximo das pontuações e, consequentemente, traduzimos tudo na escala de [0-10] normalizando os resultados.

2.2 Smartphone Based Stress Prediction

Este trabalho investiga os dados recolhidos de um *smartphone* para prever os níveis de stress de um utilizador. Foi criada uma aplicação chamada *TheStressCollector*, que uma vez instalada num Android é executada em segundo plano e recolhe periodicamente os dados de um telemóvel e carrega os para um servidor de base de dados. Os tipos de dados recolhidos foram fornecidos pelo *Google Play Services*, estes foram classificados em atividades físicas vs não-físicas. A seguir estão listadas as diferentes atividades detetadas:

- ON_FOOT: o dispositivo deteta se o utilizador esta a andar ou a correr;
- ON_BICYCLE: o dispositivo deteta se o utilizador esta numa bicicleta;
- IN_VEHICLE: o dispositivo deteta se o utilizador esta num veículo, é classificado como não-físico;
- TILTING: deteta se o ângulo do dispositivo em relação à gravidade mudou significativamente, isto é, quando o utilizador esta a utilizar o telemóvel. É classificado como não-físico;
- STILL: O dispositivo esta parado, não deteta movimento;
- UNKNOWN: Não é possível detetar a atividade atual, classificado como não-físico;

O intervalo para detetar o estado de atividade é definido em 20 minutos. Foram também verificadas as aplicações que o utilizador frequentava, não era guardado o nome das aplicações, mas sim a categoria a que pertencia (por exemplo: Saúde, Social, Entretimento, etc.). A navegação na Internet era registada e era dividido nas categorias de "Recebido" e "Enviado", "Rede" e "Rede sem Fio". Os dB das chamadas, a entrada do microfone é processada em tempo real e era medido os números dos dB durante a chamada. Após a chamada terminar os valores(dB) do mínimo, o máximo e a média são guardados e comparados. A luz do ambiente em que se encontrava o dispositivo, cada medição foi classificada com claro e escuro e só era feita a medição sempre que o utilizador interagia com o dispositivo. Era registado os horários em que as mensagens eram recebidas, sem ligar ao conteúdo. Exposição ao ruído, era registada periodicamente, ou seja, o mínimo, o máximo e a potencia media em dB de uma amostra de 20 segundos foi calculada a cada 20 minutos. E por fim a atividade do ecrã. Alem destas atividades, era feito um questionário sete vezes por dia às 8:00, 9:00, 12:00, 15:00, 18:00, 21:00 e 22:00. De manha era um questionário sobre o comportamento do sono e às 22 sobre alimentação e o stress. Ao longo do dia, eram feitos pequenos questionários sobre o consumo de alimentos e bebidas. Com toda esta recolha de dados e através de questionários foi verificado que os dados do telemóvel esta relacionado com o stress.

3 Agents Software Engineering

3.1 Goal Model

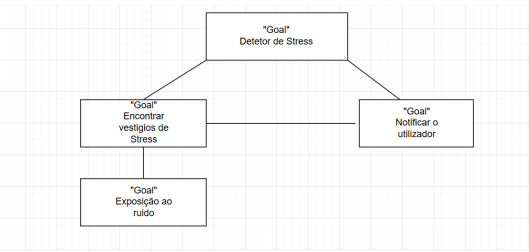

Figure 1: Goal Model

3.2 Role Model

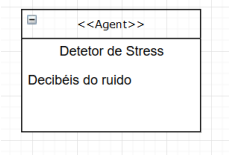

Figure 2: Role Model

4 Implementação

4.1 Sensor

Para desenvolver um sensor capaz de detetar stress foi necessário fazer pesquisa sobre o assunto para ver todo o tipo de coisas que podiam influenciar stress. Foram analisados

varias técnicas, no entanto muitas destas já estavam implementadas na aplicação Isabella. Como tal, decidiu-se procurar uma característica que não tivesse sido estudada ainda e foi aí que surgiu o ruído que uma pessoa esta envolvida. Após a escolha da característica a estudar desenvolveu-se uma aplicação capaz de guardar os decibéis do ambiente que uma pessoa se encontrava. O objetivo era guardar estes decibéis num servidor para depois poder analisar e verificar se o ruído tinha ou não influencia no stress de uma pessoa. Em primeiro lugar foi utilizada uma função que captava os decibéis através do *smartphone*, depois de captados eram convertidos, uma vez que os decibéis captados por um *smartphone* não são iguais ao que um ouvido humano é capaz de ouvir.

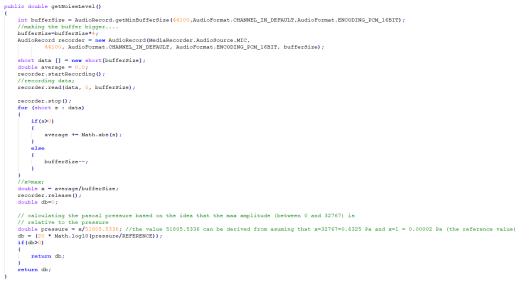

Figure 3: Função responsável por obter os decibéis

De seguida, estes decibéis são guardados num servidor, para tal foi utilizado o *Firebase* da *Google*.

Figure 4: Exemplo de captação de dB

Após esta fase e feita a recolha de dados era suposto desenvolver um modelo que fosse capaz de relacionar o ruído com o stress de uma pessoa só que surgiram alguns problemas e após falar com o professor concluiu-se que o melhor era treinar outro *dataset*. Este *dataset* irá ser apresentado na secção seguinte.

4.2 Modelo

Para o desenvolvimento desta segunda parte, foi utilizado um *dataset* resultante da captação de dados em *smartphones*. Este *dataset* denominado de *Personality Tratis Predict* capta todos os dados através do uso de aplicações. O objetivo deste é utilizar os dados de personalidade para prever o uso de aplicações. Este *dataset* é constituído por diversas *features*, no entanto a *feature* estudada no âmbito deste projeto é a *Emotional Stability*, que é a que mais se relacionava com o Stress.

Tratamento de dados

Para desenvolver este modelo, primeiramente analisou-se todas as *features* que o *dataset* continha. De seguida, a análise centrou-se na *features* mais importante, *Emotional Stability*, analisando a média, o máximo e o mínimo.

Média	-0.042369
Máximo	2.5201
Mínimo	-1.9955

Table 1: Informações relevantes sobre *EmotionalFinal*

Após esta análise conclui-se que o melhor era dividir estes valores em 3 classes:

- 1(Low)
- 2(Normal)
- 3(High)

Para atribuir estes 3 tipos estabeleceu-se três condições:

- Se o valor de *Emotional Stability* fosse inferior a -0.5 era atribuído o valor 1(Low);
- Se o valor de *Emotional Stability* fosse superior a 0.5 era atribuído o valor 3(High);
- Caso contrario, estava contido no intervalo de -0.5 a 0.5 e era atribuído o valor 2(Normal);

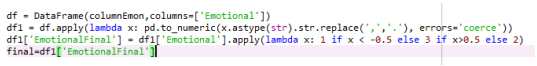

Figure 5: Excerto de código responsável por converter os dados.

Como podemos ver no excerto de código anterior, a coluna do *Emotional* é tratada e convertida para as 3 classes faladas anteriormente. Após a conversão, foi estudado as cinco *features* que apresentavam os valores mais altos de correlação com a *feature Emotional*. Com este estudo obteve-se os seguintes resultados:

Correlação	*features*
0,892686539	Carefreeness..N1.
0,869464736	Positivemood
0,845721706	Equanimity..N2.
0,736769328	Emot..robustness..N6.
0,605763535	Self.consciousness..N4.

Table 2: Correlação das *features* estudadas com a *EmotionalFinal*

Com estas *features*, fez-se o mesmo processo que foi feito anteriormente, converteu-se os valores de cada *feature* para 1(Low), 2(Normal) ou 3(High) consoante a sua média, máximo e mínimo. Tentou-se utilizar as dez *features* que mais estavam correlacionadas em vez de cinco, no entanto verificou-se que a *accuracy* baixava bastante o que indica um baixa relação com a *Emotional Stability*. Por isso decidiu-se utilizador só o top5.

Analise de dados

Nesta secção vamos analisar o *dataset* através da interpretação de gráficos. Em primeiro podemos observar como estava distribuída a idade.

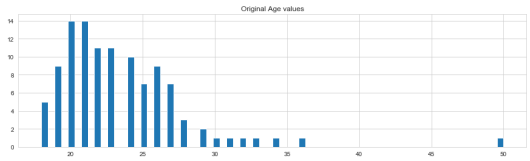

Figure 6: Dados sobre a idade do *dataset*.

Ao observamos a idade de todos os casos estudados podemos comparar esta com a *feature Emotional Stability*.

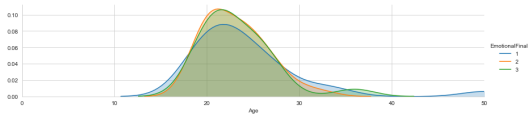

Figure 7: Relação idade *Emotional Stability*

Para comparar cada *feature* do "top5" falado anteriormente foram criados gráficos onde se consegue visualizar a sua distribuição no *dataset* assim como a sua relação à *Emotional Stability*.

Na Figura 8 verificamos o *Carefreeness*, conseguimos perceber que grande parte desta *feature* apresenta valores altos (2 e 3). Concluimos também que os valores estão relacionados com a *Emotional Stability*.

- Carefreeness..N1.

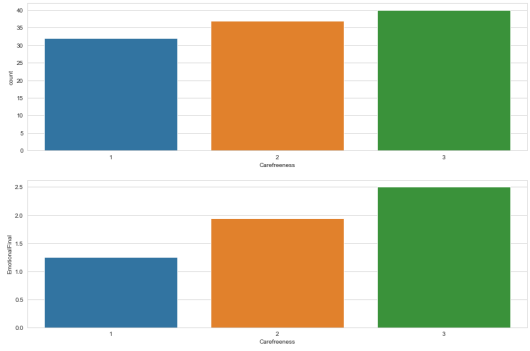

Figure 8: Dados sobre Carefreeness

Na Figura 9 visualizamos o *positivemood*, esta *feature* mostra que a maior parte dos valores tendem a pertencer à classe 3. Concluímos também que apresenta uma boa relação com *Emotional Stability.*

- Positivemood

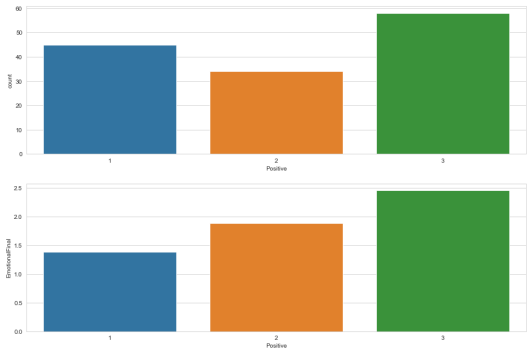

Figure 9: Dados sobre Positivemood

Na Figura 10 é analisado a *feature Equanimity N2*, nesta os valores já estão mais bem destruídos, não existe um que se destaque mais que outro e uma vez mais verificamos uma boa relação com a *Emotional Stability*.

- Equanimity..N2

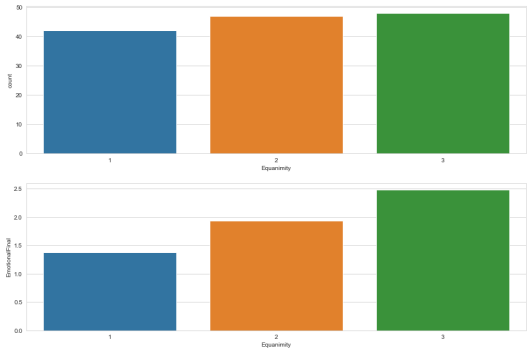

Figure 10: Dados sobre Equanimity

Na Figura 11 é a vez da *feature Emot..robustness..N6*, uma vez mais os valores estão bem distribuídos e apersentam uma boa relação.

- Emot..robustness..N6.

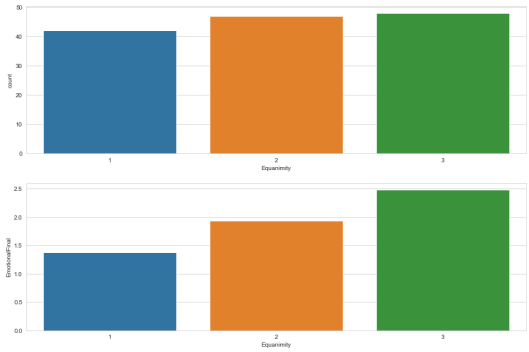

Figure 11: Dados sobre Equanimity

Por último analisamos a *feature Self.consciosness..N4* onde verificamos que a maior parte dos valores tendem a pertencer à classe 2 e sendo a *feature* com menor correlação percebe-se que a sua relação com a *Emotional Stability* não seja tão alta como as referidas anteriormente.

- Self.consciosness..N4

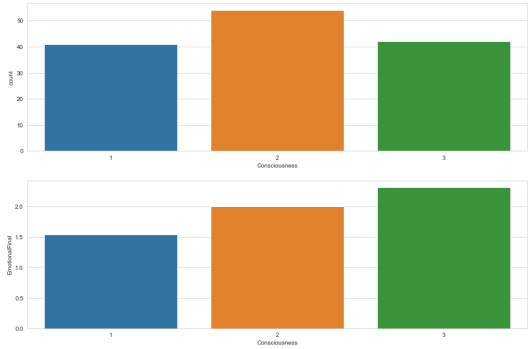

Figure 12: Dados sobre Self.consciosness

Treino

Após a análise e tratamento de dados foi desenvolvida a parte do treino. Em primeiro, dividiu-se o *dataset*, com as *features* faladas anteriormente e escolheu-se para validação 20 % como podemos ver na seguinte imagem: Estando a divisão

```
# Split-out validation dataset
array = dataset.values
X = array[:, 1:6]
Y = array[:, 0]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=seed)

# Test options and evaluation metric
seed = 7
scoring = 'accuracy'
```

Figure 13: Excerto de código sobre a preparação do treino

feita, verificou-se qual o melhor/melhores modelos a aplicar a este *dataset*, para tal utilizou-se a seguinte função:

```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)


fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

Figure 14: Excerto de código sobre a preparação do treino

Que resulta no seguinte *output*:

- LR: 0.686364 (0.173026)
- LDA: 0.842727 (0.118395)
- KNN: 0.824545 (0.120745)
- CART: 0.815455 (0.124253)
- NB: 0.708182 (0.175266)
- SVM: 0.843636 (0.129385)

Podemos concluir que o SVM e LinearDiscriminantAnalysis são os modelos que obtém melhores resultados. O seguinte gráfico mostra isso mesmo.

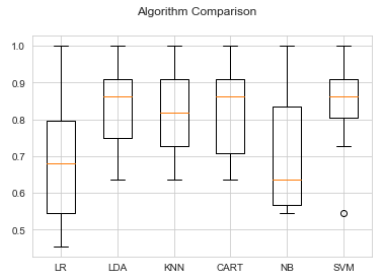

Figure 15: Comparação dos algoritmos

Modelos implementados

Decidiu-se implementar o top3 dos modelos. Para o treino de cada modelo utilizou-se as *features* faladas anteriormente. Também foi criado um *Excel* com o nome respetivo. Este *Excel* é composto por duas colunas, sendo a primeira composta por o teste e a segunda pelo "*predict*" do algoritmo.

- SVM
  Foi o modelo que apresentou melhores resultados. No excerto de código seguinte visualizamos como foi implementado.

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, Y_train)

y_pred = svclassifier.predict(X_validation)
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(Y_validation,y_pred))
print(classification_report(Y_validation,y_pred))


Submission = pd.DataFrame({"EmotinalFinalTeste":Y_validation, "EmotionalFinalPredi": y_pred })
Submission.to_csv('finalSVM.csv', index=False)
```

Figure 16: Excerto de código do SVM

O resultado para cada classe encontra-se a seguinte imagem:

	precision	recall	f1-score	support
1	0.75	1.00	0.86	6
2	0.93	0.88	0.90	16
3	1.00	0.83	0.91	6
avg / total	0.91	0.89	0.89	28

Figure 17: Precisão SVM

Quanto ao output podemos visualizar um excerto de resultados na seguinte figura.

A	B	C	D
EmotinalFinalTeste,EmotionalFinalPredi			
2,1			
2,2			
2,2			
1,1			
1,1			
2,2			
1,1			
2,1			

Figure 18: Output SVM

- *LinearDiscriminantAnalysis*
  Este modelo foi considerado o segundo melhor. No seguinte excerto de código conseguimos observar como foi implementado.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)


from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)


from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy' + str(accuracy_score(y_test, y_pred)))


Submission = pd.DataFrame({"EmotinalFinalTeste":y_test, "EmotionalFinalPredi": y_pred })
Submission.to_csv('finalLDA.csv', index=False)
```

Figure 19: Excerto de código do algoritmo LDA

Uma vez que a *accuracy* deste modelo não é tão alta como o modelo SVM não se obteve tão bons resultados, mas podemos observar alguns dos resultados na seguinte figura

17	3,2
18	2,2
19	2,3
20	3,3
21	2,2
22	3,3
23	2,2
24	3,2
25	1,2

Figure 20: Output LDA

- KNeighborsClassifier
  Este modelo foi considerado o terceiro com melhor *accuracy*. Podemos observar como foi implementado no seguinte excerto de código:

  Sendo o terceiro melhor os resultados foram menos atrativos no entanto podemos visualizar os resultados na seguinte figura

```
# Make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))


Submission = pd.DataFrame({"EmotinalFinalTeste":Y_validation, "EmotionalFinalPredi": predictions })
Submission.to_csv('finalKNN.csv', index=False)
```

Figure 21: Excerto de código do algortimo Knn

EmotinalFinalTeste,EmotionalFinalPredi			
2,1			
2,2			
2,1			
1,1			
1,1			
2,2			
1,1			
2,1			

Figure 22: Output Knn

5 Conclusão

Como podemos ver o modelo consegue acertar em grande parte das classes. Com isto podemos comprovar que as *features* que foram utilizadas para treinar o modelo tem influencia na *feature* estudade, *Emotional Stability*. Como referido em cima, se fosse utilizado dez *features* em vez de cinco os resultados não eram tão positivos, baixando a*accuracy* para cerca de 65%. Foi um estudo positivo visto acertar em muitos dos resultados, no entanto penso que com mais tempo conseguiria obter melhores resultados. Quanto à experiência ao desenvolver este trabalho foi bastante positiva pois facultou uma aprendizagem mais elaborada relativamente aos algoritmos estudados que ainda não tinha desenvolvido.