4ª Atividade de Aprendizado de Máquina
Modelos Preditivos - Parte 2

Erick Santos do Nascimento

Curso de Ciência da Computação - UECE

erick.nascimento@aluno.uece.br

Resumo

Relatório da atividade sobre modelos de classificação (parte 2) da disciplina de Aprendizado de Máquina (2022.1) da Universidade Estadual do Ceará. O objetivo é explorar as técnicas de classificação Multilayer Perceptron e Support Vector Machine.

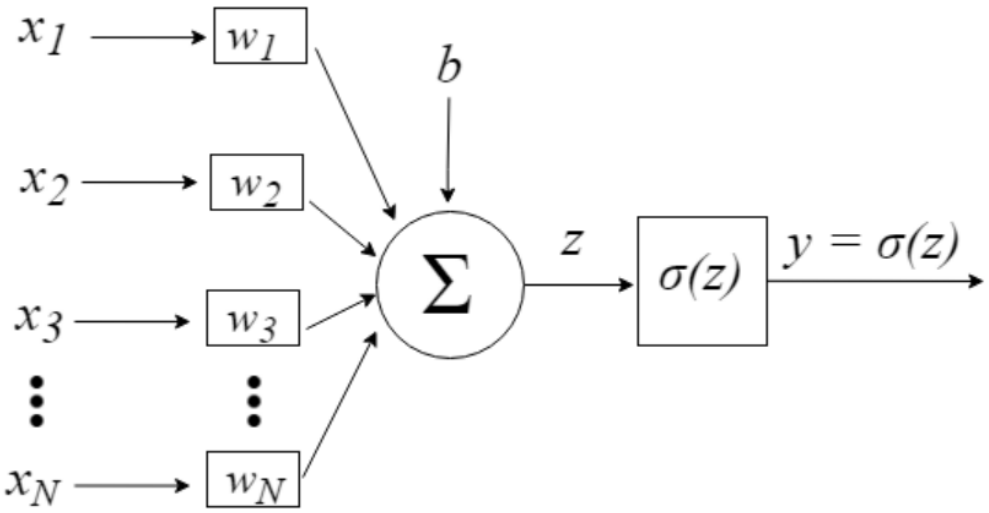
1 Introdução

Para as técnicas de **Multilayer Perceptron** e **Support Vector** Machines foram desenvolvidas as atividades:

- Fundamentação teórica sobre as técnicas
- Desenvolvimento de modelos de classificação para o dataset Water Quality Dataset com etapas de:
    - Preparação dos dados (normalização, inferir dados faltantes...)
    - Pesquisa de parâmetros
    - Treinamento e validação dos modelos
    - Avaliação dos modelos
- Comparação entre os resultados das técnicas

1.1 MLP

Inspirando-se no funcionamento dos neurônios biológicos do sistema nervoso dos animais, estabeleceu-se na área da Inteligência Artificial um modelo computacional de um neurônio conforme ilustrado a seguir:

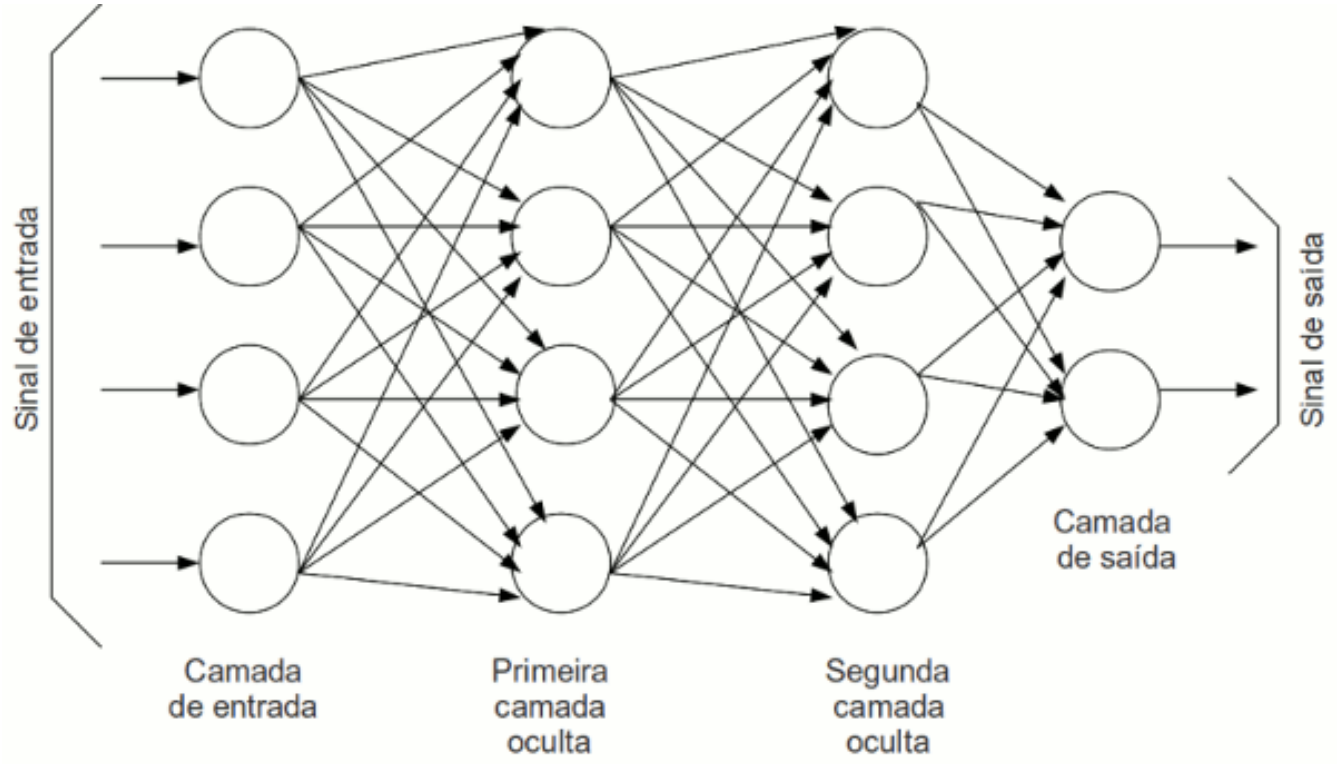


Os sinais da entrada no neurônio são representados pelo vetor x = [x1, x2, x3, …, xN], podendo corresponder aos pixels de uma imagem, por exemplo. Ao chegarem ao neurônio, são multiplicados pelos respectivos pesos sinápticos, que são os elementos do vetor w = [w1, w2, w3, …, wN], gerando o valor z, comumente denominado potencial de ativação, de acordo com a expressão:

$$z = \sum_{i=1}^{N} x_i w_i + b$$

O termo adicional b provê um grau de liberdade a mais, que não é afetado pela entrada nessa expressão, correspondendo tipicamente ao “bias” (viés). O valor z passa então por uma função matemática de ativação σ, com a característica de ser não linear, responsável por limitar tal valor a um certo intervalo, produzindo o valor final de saída y do neurônio. Algumas funções de ativação usadas são a degrau, sigmóide, tangente hiperbólica, softmax e ReLU (Rectified Linear Unit).

Com apenas um neurônio não se pode fazer muita coisa, mas podemos combiná-los em uma estrutura em camadas, cada uma com número diferente de neurônios, formando uma rede neural denominada Perceptron Multicamadas (“Multi Layer Perceptron — MLP”). O vetor de valores de entrada x passa pela camada inicial, cujos valores de saída são ligados às entradas da camada seguinte, e assim por diante, até a rede fornecer como resultado os valores de saída da última camada. Pode-se arranjar a rede em várias camadas, tornando-a profunda e capaz de aprender relações cada vez mais complexas.



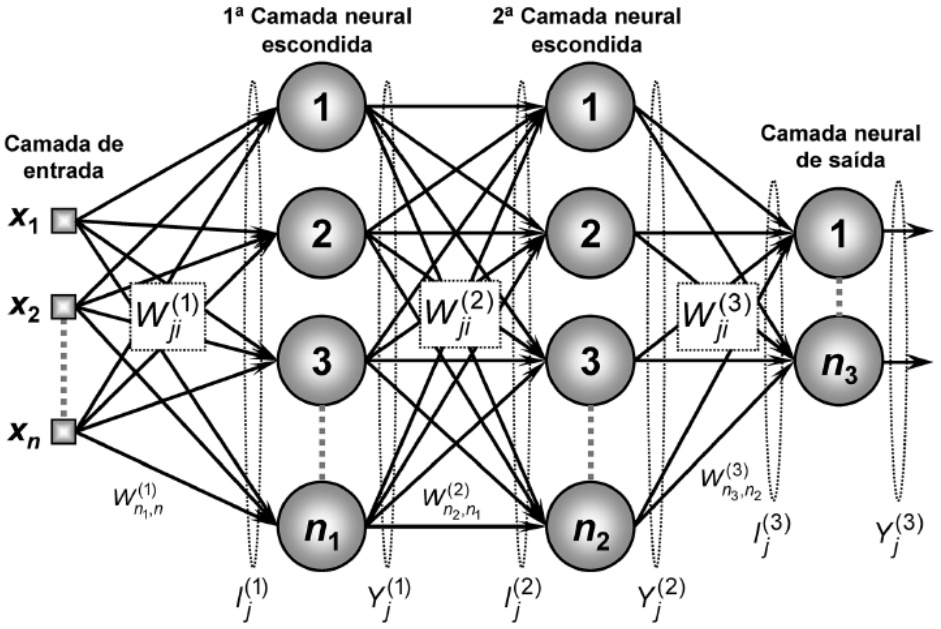
Para que uma rede dessas funcione, é preciso treiná-la. O treinamento de uma rede MLP insere-se no contexto de aprendizado de máquina supervisionado, em que cada amostra de dados utilizada apresenta um rótulo informando a que classificação ela se encaixa. Por exemplo, uma imagem de um cachorro contém um rótulo informando que aquilo é um cachorro. Assim, a ideia geral é fazer com que a rede aprenda os padrões referentes a cada tipo de coisa (cada classe), assim, quando uma amostra desconhecida for fornecida à rede, ela seja capaz de estabelecer a qual classe tal amostra pertence. Como isso pode ser feito?

A ideia do algoritmo backpropagation é, com base no cálculo do erro ocorrido na camada de saída da rede neural, recalcular o valor dos pesos do vetor w da camada última camada de neurônios e assim proceder para as camadas anteriores, de trás para a frente, ou seja, atualizar todos os pesos w das camadas a partir da última até atingir a camada de entrada da rede, para isso realizando a retropropagação o erro obtido pela rede. Em outras palavras, calcula-se o erro entre o que a rede

achou que era e o que de fato era (era um gato e ela achou que era um cachorro — erro!), então recalculamos o valor de todos os pesos, começando da última camada e indo até a primeira, sempre tendo em vista diminuir esse erro.

Na camada de saída se usa os labels fornecidos pelos dados e nas camada internas se usa a regra delta generalizada:

Em uma rede de 3 camadas por exemplo:



A atualizações dos pesos ficam:

Camada de saida:

$$W_{ji}^{(3)} \leftarrow W_{ji}^{(3)} + \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \qquad \delta_j^{(3)} = (d_j - Y_j^{(3)}) \cdot g'(I_j^{(3)})$$

Segunda camada:

$$W_{ji}^{(2)} \leftarrow W_{ji}^{(2)} + \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \qquad \delta_j^{(2)} = -(\sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)}) \cdot g'(I_j^{(2)})$$

Primeira camada:

$$W_{ji}^{(1)} \leftarrow W_{ji}^{(1)} + \eta \cdot \delta_j^{(1)} \cdot x_i \qquad \delta_j^{(1)} = -(\sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)}) \cdot g'(I_j^{(1)})$$

Onde:

- $W_{ji}^{(L)}$ são matrizes de pesos cujos elementos denotam o valor do peso sináptico conectando o j-ésimo neurônio da camada (L) ao i-ésimo neurônio da camada (L-1).
- $I_j^{(L)}$ são vetores cujos elementos denotam a entrada ponderada em relação ao j-ésimo neurônio da camada L.
- d_j: é a saída esperada
- $Y_j^{(L)}$: são vetores cujos elementos denotam a saída do j-ésimo neurônio em relação à camada L.
- E g(x) a função de ativação e g´(x) sua derivada

Ajusta os pesos dos neurônios da camada de saída da rede levando-se em consideração a diferença observada entre as respostas produzidas por suas saídas em relação aos seus respectivos valores desejados.

Em **resumo**, o backpropagation consiste nas seguintes etapas:

1. Inicialize todos os pesos da rede com pequenos valores aleatórios.
2. Fornecer dados de entrada à rede e calcular o valor da função de erro obtida, ao comparar com o valor de saída esperado. Lembre-se de que como o aprendizado é supervisionado, já se sabe de antemão qual deveria ser a resposta correta (na camada de saída se usa os labels, nas camadas escondidas se usa a regra delta generalizada). É importante que a **função de erro seja diferenciável** para que o cálculo do gradiente possa ser feito.
3. Na tentativa de minimizar o valor da função de erro, calculam-se os valores dos gradientes para cada peso da rede. Do Cálculo, sabemos que o vetor gradiente fornece a direção de maior crescimento de uma função; aqui, como queremos caminhar com os pesos na direção de maior decréscimo da função de erro, basta tomarmos o sentido contrário ao do gradiente.
4. Uma vez que temos o vetor gradiente calculado, atualizamos cada peso de modo iterativo, sempre recalculando os gradientes em cada passo de iteração, até o erro diminuir e chegar abaixo de algum limiar preestabelecido, ou o número de iterações atingir um valor máximo, quando enfim o algoritmo termina e a rede está treinada.

Assim, a fórmula geral de atualização dos pesos na iteração fica:

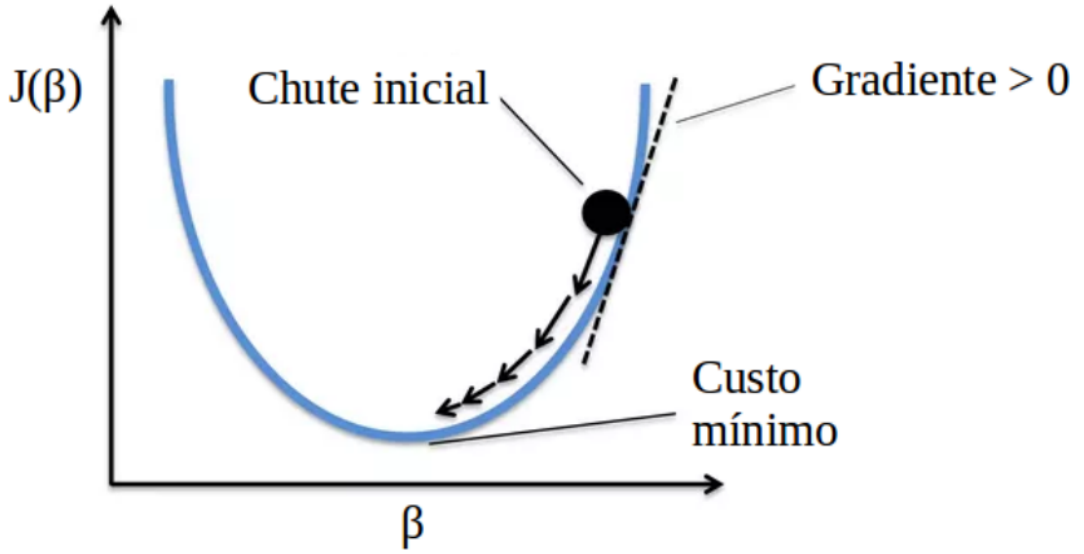
$$w \leftarrow w - \eta \frac{\partial E}{\partial w}$$

Ou seja, o valor do peso na iteração atual será o valor do peso na iteração anterior, corrigido de valor proporcional ao gradiente. O sinal negativo indica que estamos indo na direção contrária à do gradiente, conforme mencionado. O parâmetro η representa a taxa de aprendizado da rede neural, controlando o tamanho do passo que tomamos na correção do peso.

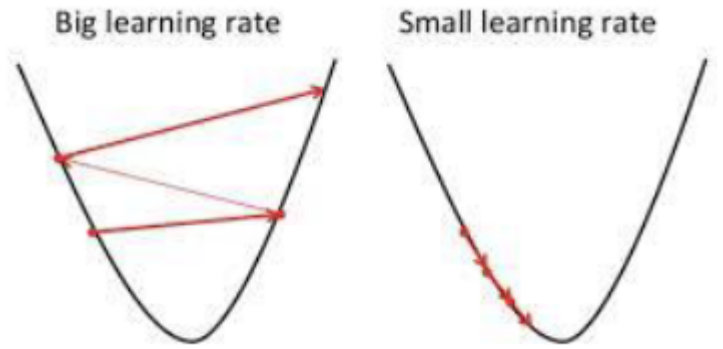
Essa forma de otimização é conhecida como *gradiente descendente*

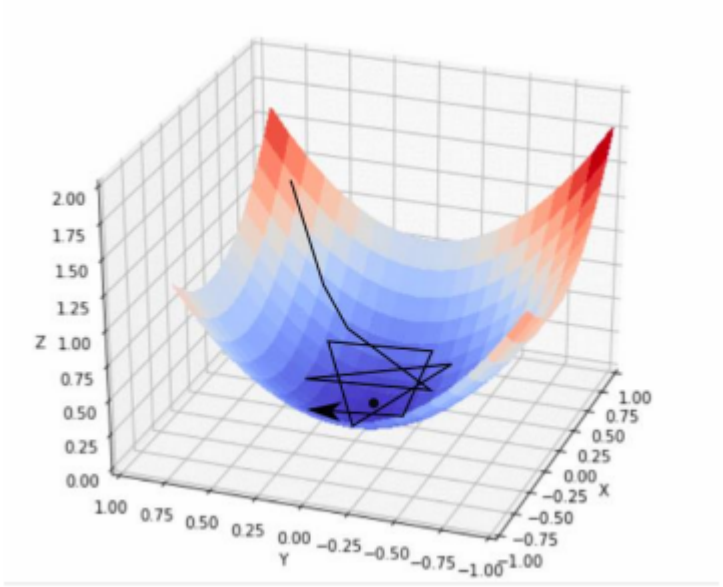
Gradiente descendente

Gradiente descendente é um dos algoritmos de maior sucesso em problemas de Machine Learning. O método consiste em encontrar, de forma iterativa, os valores dos parâmetros que minimizam determinada função de interesse. No nosso caso a função é f(w) = E, onde w são os pesos da rede e E é o erro da rede em comparação com as amostras de treinos.



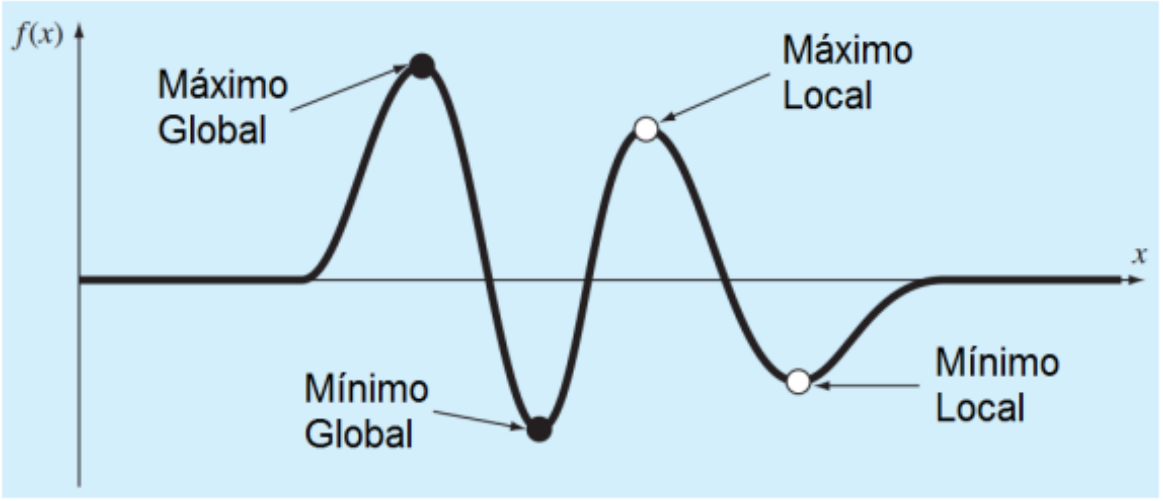
A taxa de aprendizagem e o termo momentum tem funções parecidas, ter um certo controle no tamanho do passo em direção a o declínio da função, a diferença é que a taxa de aprendizagem é constante e o momentum é variável e tende a diminuir quando as épocas passam, uma taxa de aprendizagem baixa leva a passos menores e a uma descida mais controlada e “precisa” de forma a sempre encontrar erros menores quando as épocas passam, mais por dar passos menores a convergência para o mínimo pode demorar bastante, já uma taxa maior converge mais rapidamente mas os passos são maiores gerando uma descida menos controlada fazendo com que as vezes o erro aumenta de uma época para outra e em alguns casos perdendo o mínimo local e seguindo em direção a outro, por isso também uma taxa de aprendizagem grande também dificulta a precisão já que dependendo da superfície da função ao se aproximar do mínimo fica mais difícil acertar pontos com erros menores. O termo momentum tem um efeito mais otimizado, gerando passos maiores no início e menores com o passar das épocas, tendendo a zero e deixando apenas o efeito da taxa de aprendizagem. Logo observo que no início o momentum deve controlar o tamanho do passo e ao final quando o momentum tender a zero uma pequena taxa de aprendizagem deve guiar o processo de aprendizagem ao mínimo local.





Ótimo local vs Ótimo global

O ótimo global representa a melhor solução, enquanto que um ótimo local representa a melhor solução de uma vizinhança. Como no gradiente descendente sempre se desce é possível “cair e ficar preso” no ótimo local. Normalmente estamos interessados em descobrir o ótimo global ou seja o local de menor erro, mais é comum achar um ótimo local e supor que ele seja um ótimo global, mas é extremamente dificil saber se esta em um otimo local ou em um ótimo global, existem formas de se evitar um ótimos locais uma delas é a inicialização aleatória dos pesos e várias execuções.



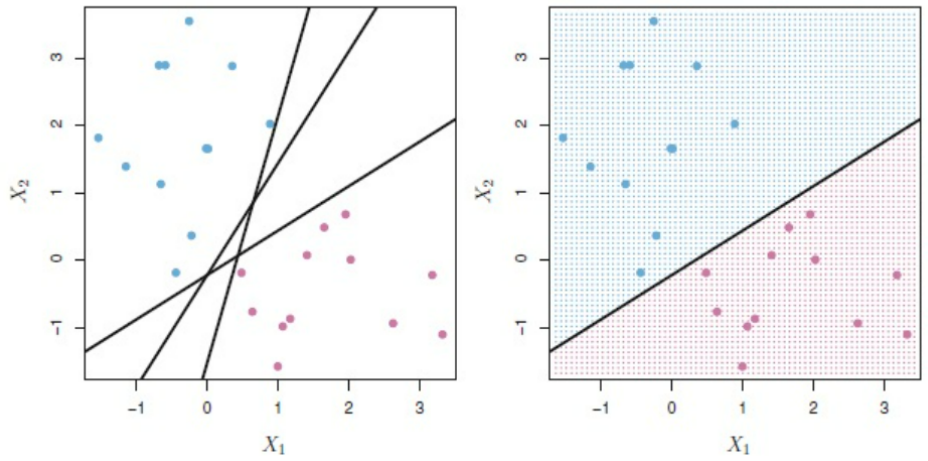
Assim MLP pode ser visto teoricamente como um aproximador universal com alto poder de discriminação e generalização mais tendem a ser muito complexos e custosos no treino.

1.2 SVM

Maximal Margin Classifier

O Maximal Margin Classifier (ou Classificador da Margem Máxima) é o método utilizado em problemas de classificação, no qual seja possível utilizar um hiperplano para separar as observações baseadas no aspecto desejado. No caso de uma observação com 2 variáveis (ou seja, um plano bidimensional), o hiperplano será uma reta (espaço unidimensional). Portanto, os dados precisam ser totalmente separáveis. Como existe uma infinidade de posições possíveis para o hiperplano caso os dados sejam perfeitamente separáveis, esse classificador busca determinar essa posição a partir da maximização da margem.

Na imagem abaixo à esquerda, existem duas classes de observações, rosa e azul, cada uma delas possuem medidas de duas variáveis. Três hiperplanos possível, dentre vários, são mostrados em preto.À direita, um hiperplano é utilizado como um classificador.



A margem diz respeito ao espaço vazio que fica entre as observações do extremo interno de cada grupo (ou seja, as observações da “ponta” mais próxima ao outro grupo) e o hiperplano.

Como neste classificador a margem é intransponível, utiliza-se otimização para alargá-la o máximo possível a fim de criar um espaço de segurança para as classificações.

Assim:

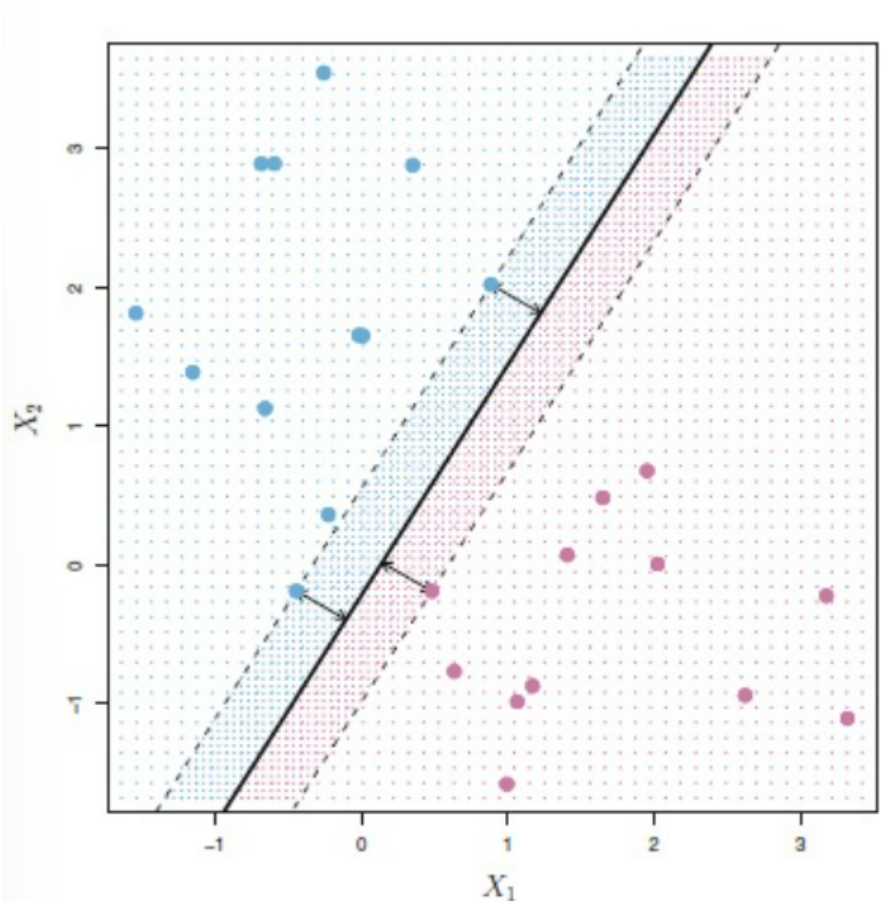
$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n.$$

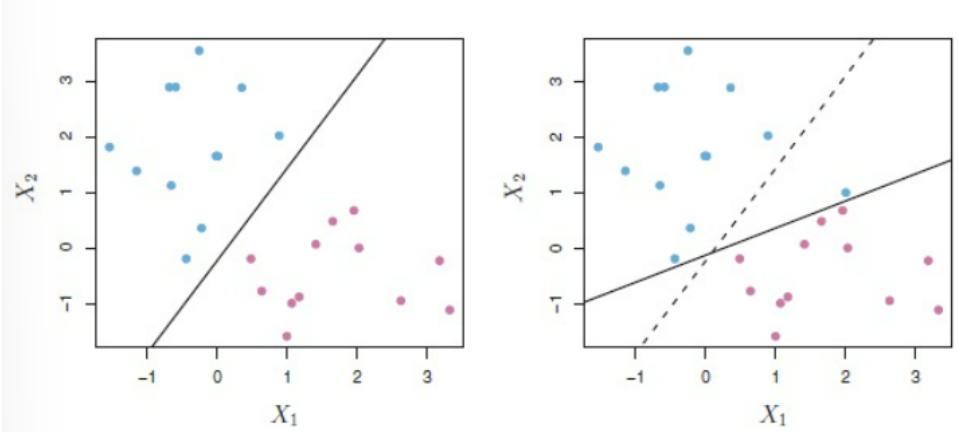
As restrições asseguram que cada observação estará no lado correto do hiperplano e, no mínimo, a uma distância M do mesmo. Consequentemente, M representa a margem, e o problema de otimização escolhe os β s para maximizar M. Essa é a exata definição de margem máxima.

Na imagem existem duas classes de observações, rosa e azul. O hiperplano ótimo é representado pela linha sólida. A margem é a distância entre a linha sólida e a linha pontilhada de cada lado.



Porém, caso exista algum outlier no conjunto de dados ou os grupos não sejam completamente separáveis, esse método perde muito em eficiência ou até a aplicação, pois ou a margem fica pequena demais ou não é possível traçar o hiperplano.

Na imagem à esquerda, duas classes de observações são classificadas por um hiperplano ótimo. À direita, uma observação em azul foi acrescentada, causando uma mudança dramática na escolha do hiperplano ótimo, da reta pontilhada para a negritada.



Support Vector Classifier

O Support Vector Classifier (ou Classificador dos Vetores de Suporte) é uma generalização do modelo anterior, visto que se aplica aos casos em que o Maximal Margin Classifier não se aplica. Isso se dá pela flexibilização em relação à invasão da margem por parte de determinadas observações. Esse classificador, ao permitir isso, busca um melhor mecanismo de classificação no longo prazo (permitindo a classificação “errada” de certas observações). Portanto, o problema de otimização do classificador anterior ganha uma nova forma, com um parâmetro que controlará o “grau” de permissão de invasão à margem.

Assim:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

onde C é a um parâmetro não-negativo de ajustamento. Como no problema de otimização anterior, M é a variável que representa a margem, que procuramos maximizar. ε s são variáveis de erro que permitem observações com problemas de classificação.

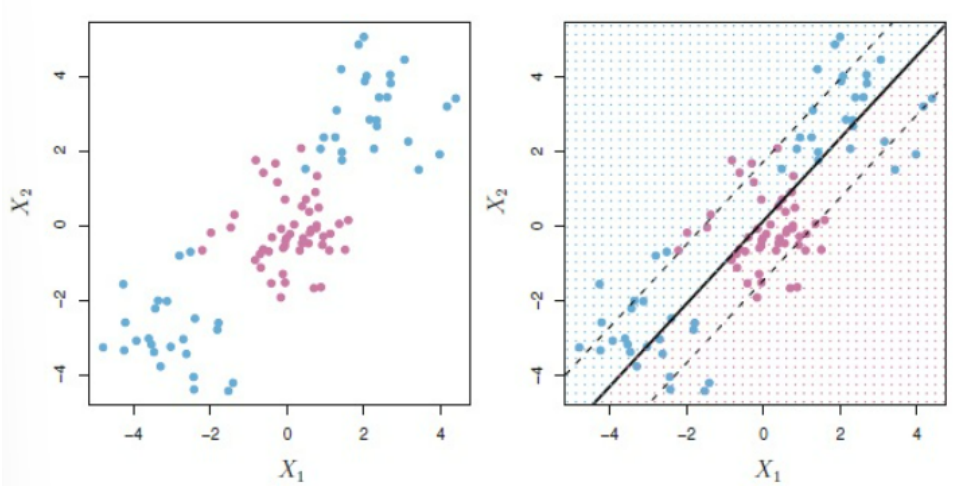
Após a resolução do problema, classificamos uma observação teste, x^*a ao determinar de qual lado do hiperplano ela ficará. Ou seja, classificamos a observação com base no sinal da seguinte equação:

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \ldots + \beta_p x_p^*$$

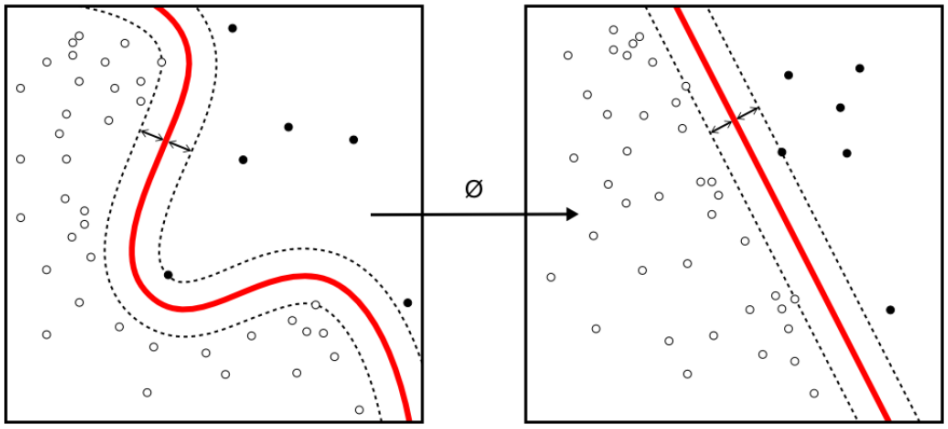
A partir disso, observa-se a aplicação dos dois conceitos estruturantes: o modelo permitirá que algumas observações sejam classificadas de maneira errada (com um maior viés) para que a classificação no longo prazo seja melhor (uma menor variância). O parâmetro que controla essa permissão, C, é determinado por validação cruzada, achando o ponto ótimo de equilíbrio entre essas duas características.

Support Vector Machine (SVM)

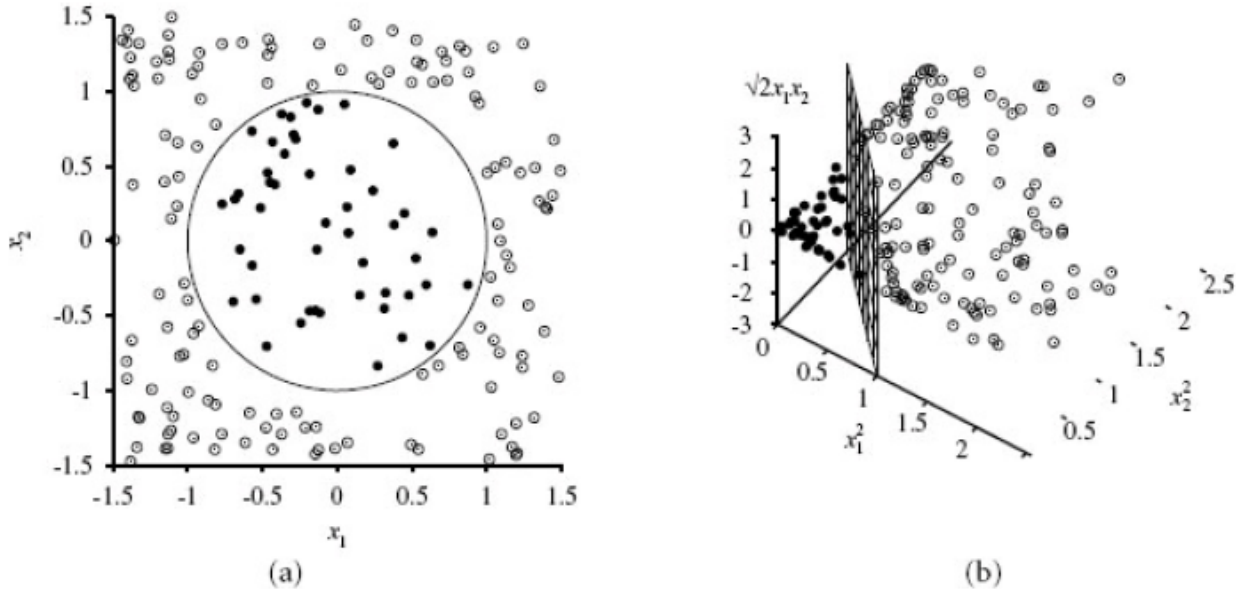
No entanto, existem grupos que não são linearmente separáveis, como os da imagem abaixo.



Para separar esses tipos de exemplos, o algoritmo primeiro faz uma transformação não-linear do espaço para fazê-los linearmente separáveis. Dessa forma, apesar da separação ser um hiperplano no espaço das features (como chamamos o espaço depois da transformação), no espaço das entradas (como chamamos o espaço inicial) a separação é não-linear. A função usada na transformação é a função kernel



Outro exemplo



(a) Um conjunto de treinamento em duas dimensões com exemplos positivos como círculos pretos e exemplos negativos como círculos brancos. A fronteira de decisão verdadeira também é exibida, $x_1^2 + x_2^2 \leq 1$.

(b) Os mesmos dados após o mapeamento em um espaço de entrada tridimensional

$x_1^2, x_2^2, \sqrt{2}x_1x_2$. A fronteira de decisão circular em (a) torna-se uma fronteira de decisão linear em três dimensões.

Quando o Support Vector Classifier é combinado com uma função Kernel, como a polinomial, o classificador resultante é conhecido como Support Vector Machine.

Esse tipo de modelo se destaca na classificação de dados espalhados de maneira não regular, já que a separação não precisa ser linear e nem a mesma para todos os dados. Esse algoritmo é muito interessante para iniciantes também porque não é necessário tanto conhecimento da base de dados para conseguir uma predição com boa acurácia. Além disso, o SVM funciona bem em espaços com muitas dimensões (muitas features) e é garantido a convergir para o melhor hiperplano possível, visto que seu algoritmo não se perde em mínimos locais como acontece com redes neurais (que veremos mais adiante). Contudo, o resultado do SVM é dificilmente interpretável (mas possível) e, conforme o tamanho do dataset vai aumentando, o tempo necessário para fazer os cálculos cresce muito rapidamente e a interpretabilidade cai mais rápido ainda.

2 Metodologia

Utilizando o dataset Water Quality Dataset, as seguintes atividades foram feitas:

1. Os valores não válidos foram preenchidos com o algoritmo KNN
2. Os dados de entrada foram normalizados

Com MLP:

1. Com todos os dados foi feito uma pesquisa para os valores de:
    1.1. Taxa de aprendizagem: 0.01 a 1 com passo de 0.1
    1.2. Momentum: 0.01 a 1 com passo de 0.05
2. Com a melhor combinação taxa de aprendizagem X momentum (baseado principalmente na acurácia) foi feito o cross-validation com a melhor combinação com K=10
3. Avaliação dos modelos, (acurácia, precisão, recall, f1, curva roc e matriz de confusão)

Com SVM:

1) Com todos os dados foi feito uma pesquisa para os valores de C, Gamma e graus, utilizando alguns tipos de kernel:
    a) Linear
    b) RBF
    c) Polinomial
2) Com a melhor combinação foi feito o cross-validation com a melhor combinação com K=10
3) Nova tentativa com novos parâmetros e outra função
4) Avaliação dos modelos, (acurácia, precisão, recall, f1, curva roc e matriz de confusão)

3 Resultados

acc = acurácia

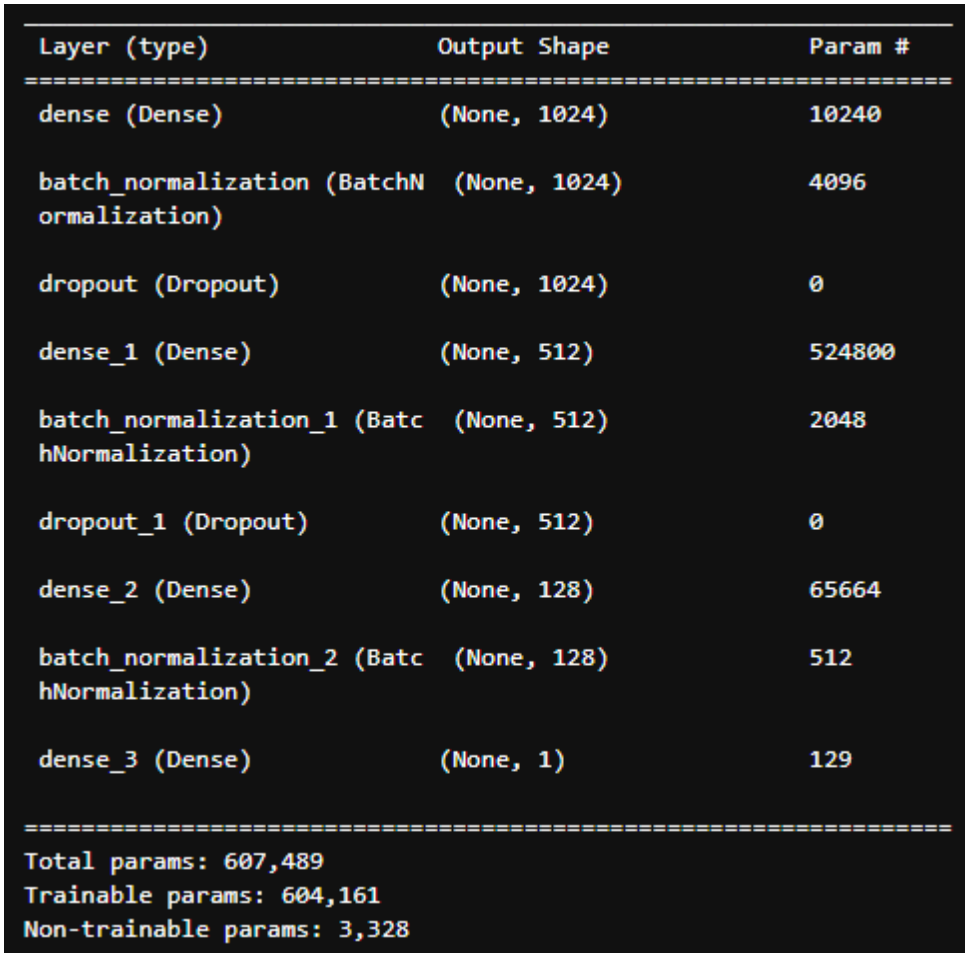
pre = precisão

rec = recall

f1 = f1-score

MLP

Arquitetura da rede



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1024)	10240
batch_normalization (BatchNormalization)	(None, 1024)	4096
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
batch_normalization_1 (BatchNormalization)	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
batch_normalization_2 (BatchNormalization)	(None, 128)	512
dense_3 (Dense)	(None, 1)	129

Total params: 607,489
Trainable params: 604,161
Non-trainable params: 3,328

Depois de pesquisar uma boa combinação de taxa de aprendizagem X momentum foi gerado o seguinte resultado:

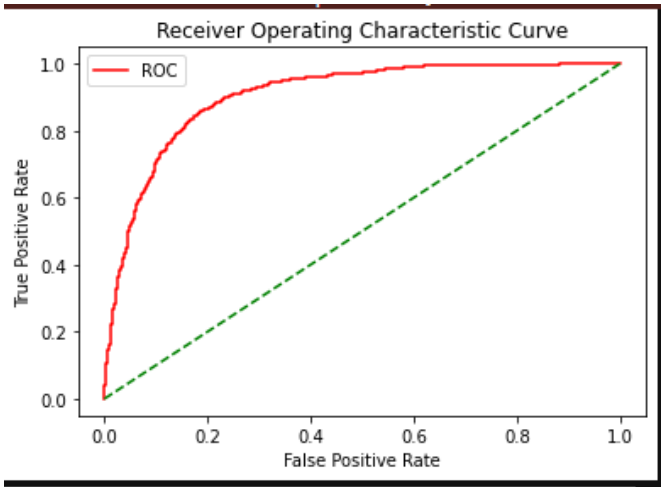
Melhores combinações com a média de 5 execuções (devido a natureza aleatória da MLP) e todos os dados

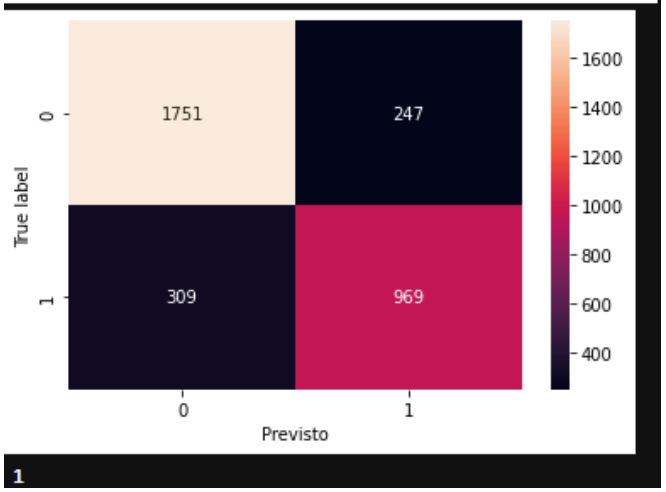
```
['loss', 'acc', 'learning_rate', 'momentum']
[0.28170503973960875, 0.9949278950691223, 0.01, 0.36],
[0.23264294266700744, 0.9923421144485474, 0.01, 0.16],
[0.25377528071403505, 0.991745400428772, 0.01, 0.26],
[0.27415605783462527, 0.9916459441184997, 0.01, 0.31],
[0.3456865906715393, 0.9914470434188842, 0.01, 0.51],
[0.2476332813501358, 0.9909497737884522, 0.01, 0.21],
[0.3276894807815552, 0.9907508611679077, 0.01, 0.46],
[0.3051791965961456, 0.9896568894386292, 0.01, 0.41],
[0.3648068308830261, 0.987468934059143, 0.01, 0.56],
[0.3889909446239471, 0.9871705770492554, 0.01, 0.61],
[0.24908385574817657, 0.9863749504089355, 0.01, 0.06],
[0.4232665359973907, 0.9818995594978333, 0.01, 0.66],
[0.25232577323913574, 0.9812033772468567, 0.01, 0.11],
...
```

Usando os melhores parâmetros foi feito um cross-validation com a melhor combinação com K=10, resultando nos seguintes resultados.

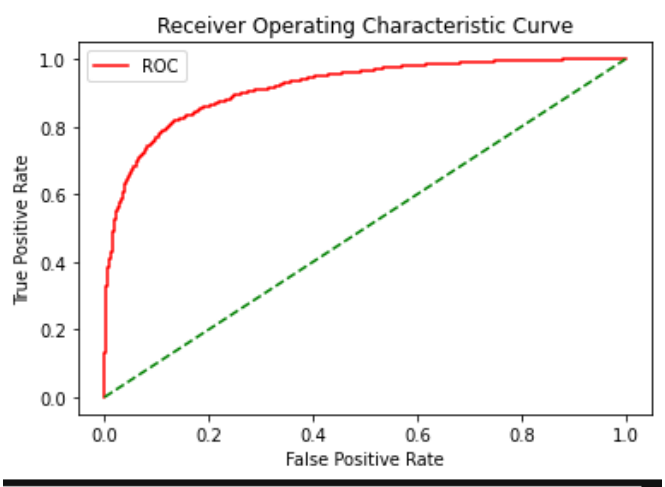
K-Fold	train_acc	val_acc	train_rec	val_rec	train_pre	val_pre	train_f1	val_f1
1	0.8514246947082768	0.6402439024390244	0.7945804195804196	0.44776119402985076	0.8174460431654677	0.5769230769230769	0.8058510638297873	0.5042016806722689
2	0.8483717774762551	0.8536585365853658	0.7450980392156863	0.7142857142857143	0.8551859099804305	0.8064516129032258	0.7963553530751708	0.7575757575757576
3	0.89280868385346	0.801829268292683	0.8218940052128584	0.6929133858267716	0.8949858088930936	0.7719298245614035	0.8568840579710145	0.7302904564315352
4	0.8578697421981004	0.7774390243902439	0.8019197207678883	0.7121212121212122	0.8271827182718272	0.7286821705426356	0.8143553389455029	0.7203065134099617
5	0.8639755766621439	0.7774390243902439	0.76137339055794	0.5929203539823009	0.8782178217821782	0.7127659574468085	0.815632183908046	0.6473429951690822
6	0.9073948439620081	0.7957317073170732	0.8811002661934338	0.7350993377483444	0.877208480565371	0.8043478260869565	0.8791500664010624	0.7681660899653979
7	0.8467277043065445	0.8287461773700305	0.7706342311033884	0.7559055118110236	0.8251162790697675	0.7933884297520661	0.7969451931716082	0.7741935483870968
8	0.8701254662597491	0.8868501529051988	0.8079930495221547	0.8346456692913385	0.8516483516483516	0.8688524590163934	0.8292465448060634	0.8514056224899599
9	0.8745337402509326	0.8470948012232415	0.8227628149435273	0.7952755905511811	0.8508535489667565	0.808	0.8365724381625442	0.8015873015873016
10	0.880637504238725	0.8440366972477065	0.8915135608048994	0.8074074074074075	0.8171611868484362	0.8134328358208955	0.8527196652719665	0.8104089219330854

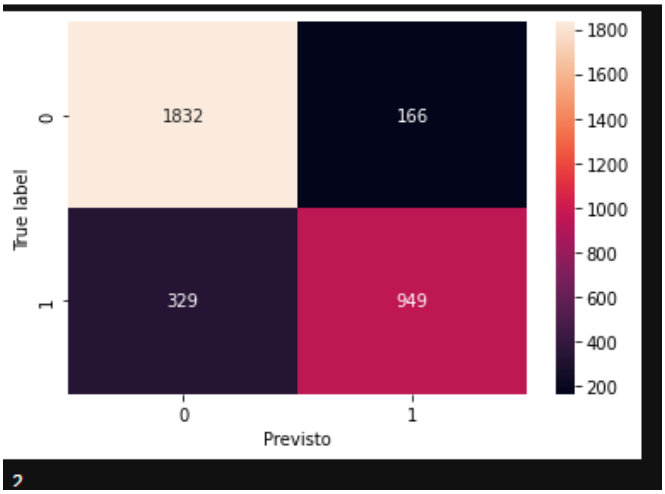
E nas seguintes curvas ROC e matrizes de confusão:


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

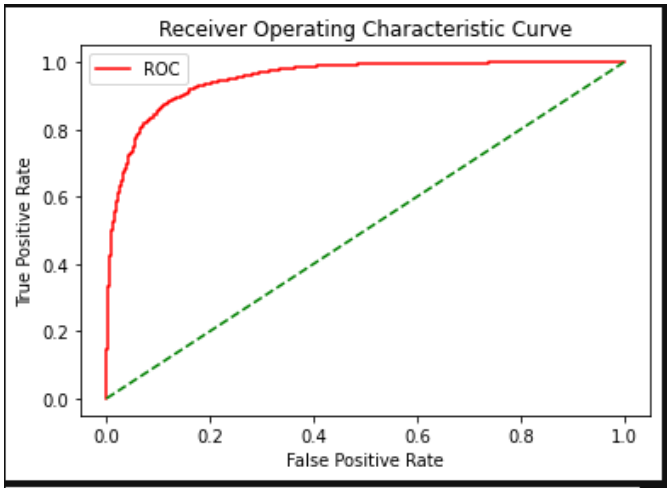

1751
247
309
969
True label
Previsto
0
1
1600
1400
1200
1000
800
600
400

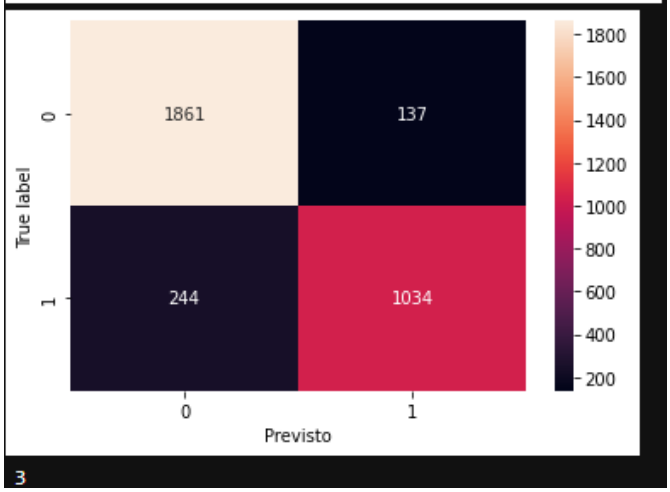
1


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

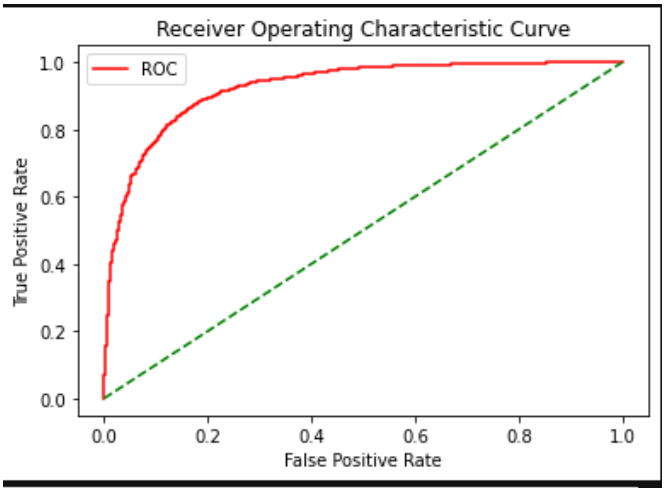

1832
166
329
949
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

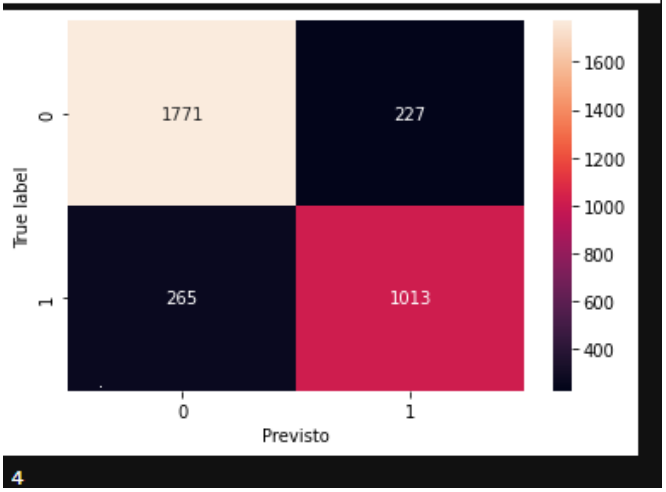
2


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

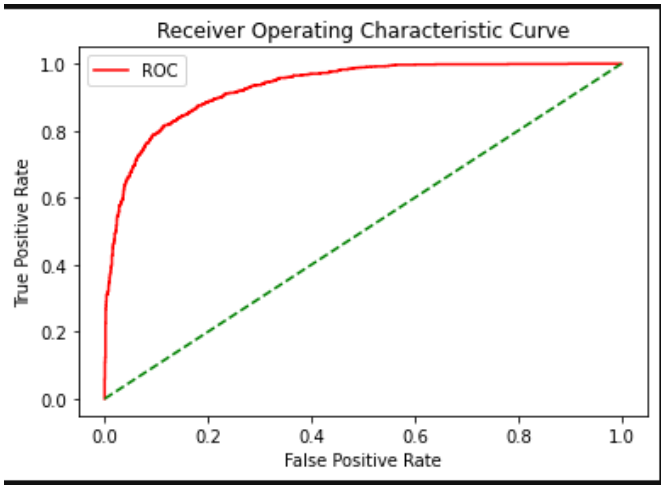

1861
137
244
1034
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

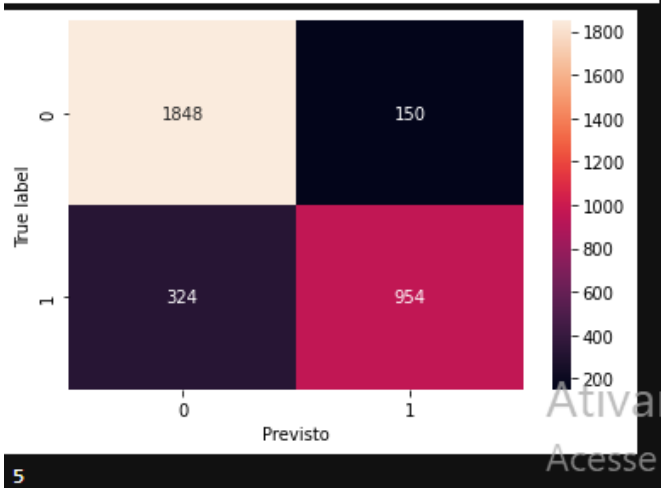
3


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

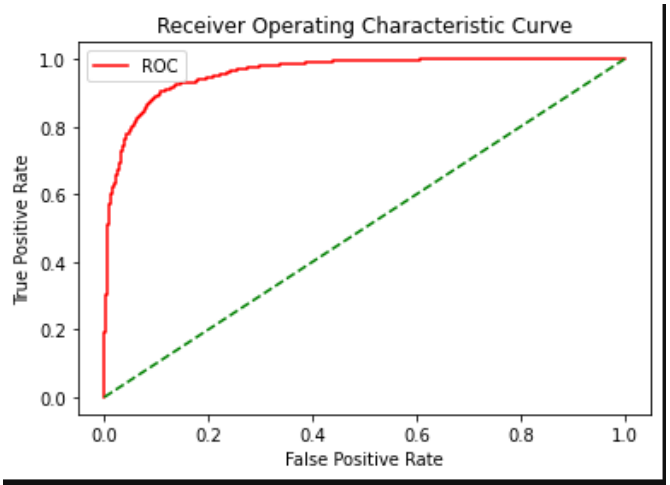

1771
227
265
1013
True label
Previsto
0
1
1600
1400
1200
1000
800
600
400

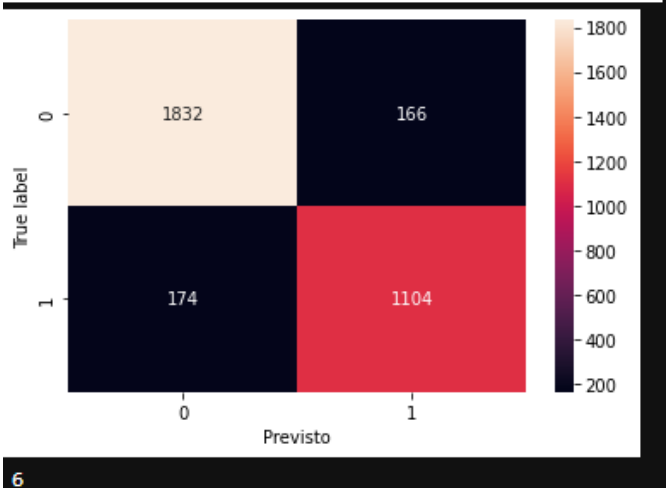
4


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

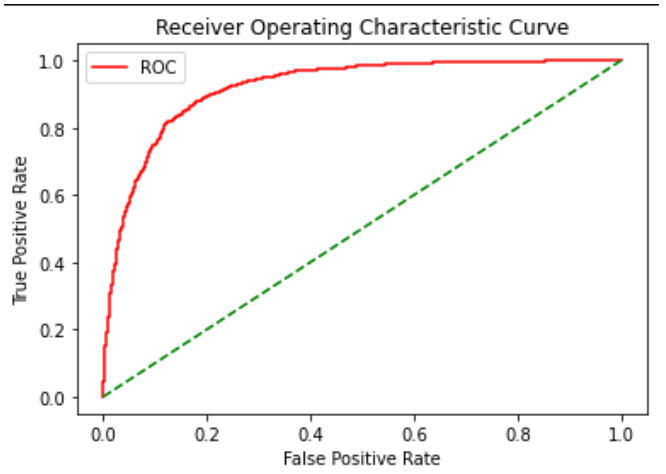

1848
150
324
954
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

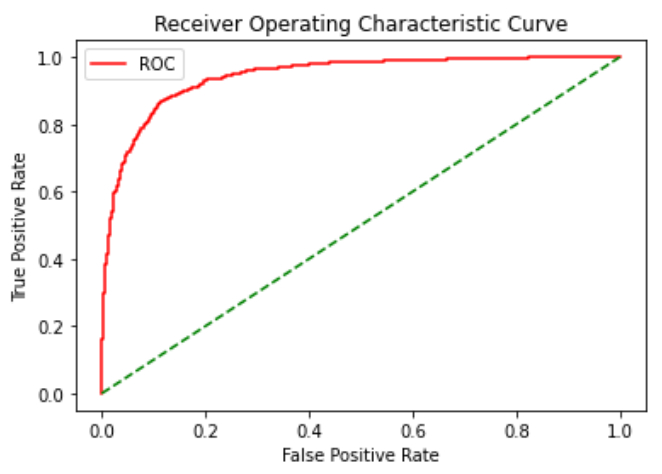
5

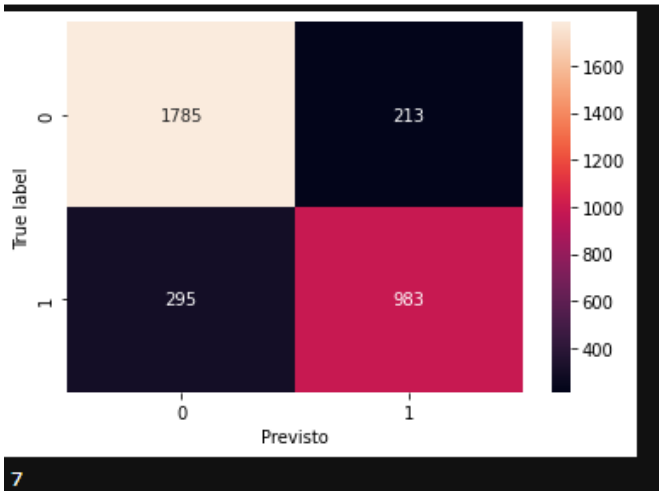

Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


1832
166
174
1104
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

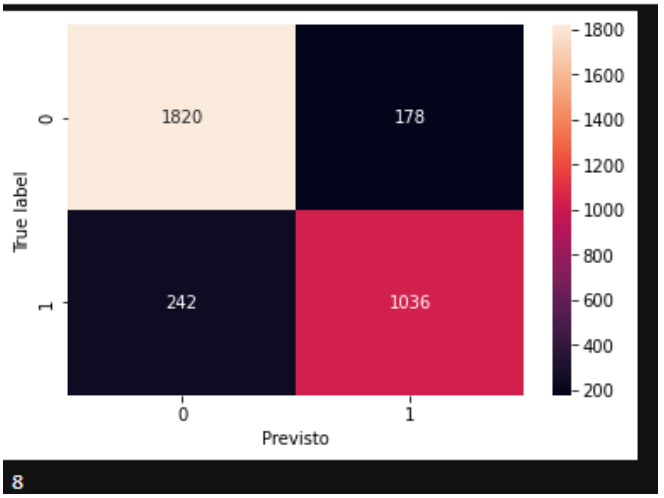
6


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

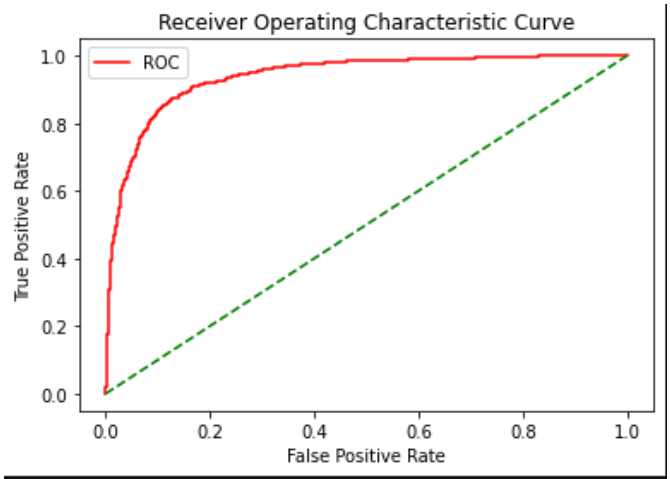

Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

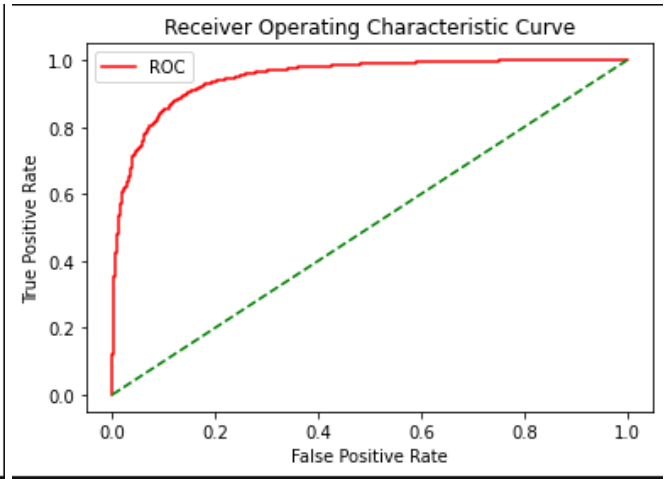

1785
213
295
983
True label
Previsto
0
1
1600
1400
1200
1000
800
600
400

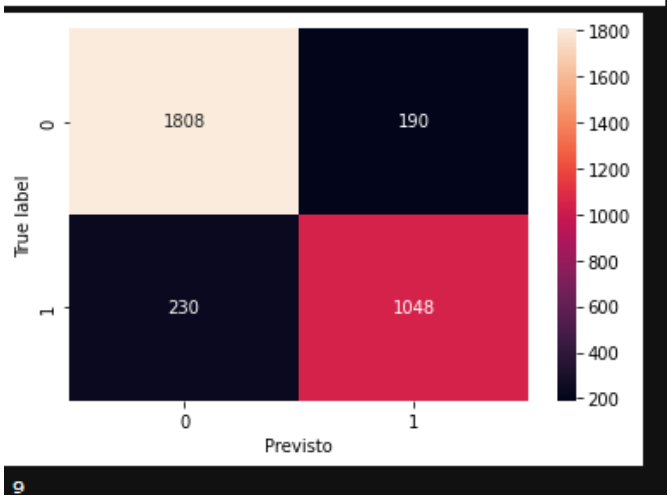
7


1820
178
242
1036
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

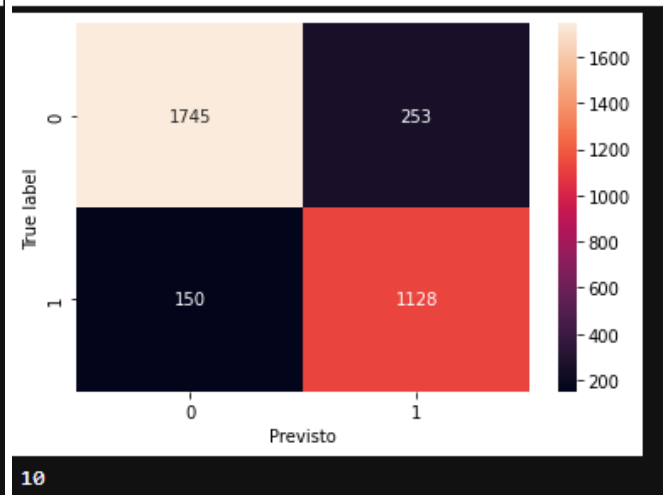
8


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


1808
190
230
1048
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

9


1745
253
150
1128
True label
Previsto
0
1
1600
1400
1200
1000
800
600
400
200

10

SVM

Também foi realizada uma pesquisa para achar os melhores parâmetros, para cada função com os melhores resultados, (a seguir apenas os melhores para cada função):

Linear

['acc', 'C']

[0.6098901098901099, 0.01],

[0.6098901098901099, 0.03],

[0.6098901098901099, 0.05],

[0.6098901098901099, 0.07],

[0.6098901098901099, 0.09],

[0.6098901098901099, 0.11],

[0.6098901098901099, 0.13],

[0.6098901098901099, 0.15],

...

RBF

['acc', 'gamma', 'C']

[0.7289377289377289, 'scale', 0.99],

[0.7283272283272283, 'scale', 0.95],

[0.7283272283272283, 'scale', 0.97],

[0.7268009768009768, 'scale', 0.93],

[0.7261904761904762, 'scale', 0.91],

[0.7258852258852259, 'scale', 0.89],

[0.7258852258852259, 'auto', 0.97],

[0.7252747252747253, 'auto', 0.95],

[0.724053724053724, 'scale', 0.87],

[0.7237484737484737, 'auto', 0.93],

[0.7231379731379731, 'auto', 0.91],

[0.7228327228327228, 'scale', 0.85],

[0.7213064713064713, 'auto', 0.89],

[0.721001221001221, 'scale', 0.83],

[0.7206959706959707, 'auto', 0.87],

[0.7191697191697192, 'scale', 0.81],

[0.7191697191697192, 'auto', 0.85],

[0.717948717948718, 'scale', 0.79],

[0.7176434676434676, 'auto', 0.81],

...

Polynomial

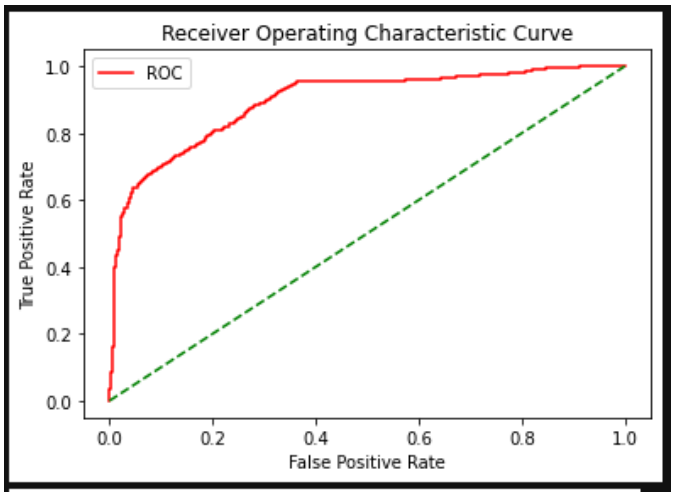
['acc', 'gamma', 'C', 'degree']

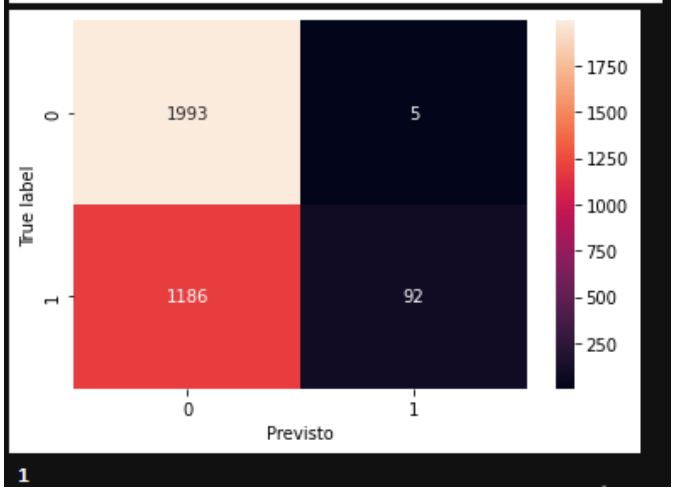
```
[[0.7872405372405372, 'scale', 0.99, 10],
 [0.7872405372405372, 'scale', 0.99, 12],
 [0.7869352869352869, 'scale', 0.99, 14],
 [0.7866300366300366, 'scale', 0.97, 10],
 [0.7866300366300366, 'scale', 0.97, 14],
 [0.7863247863247863, 'scale', 0.93, 10],
 [0.7863247863247863, 'scale', 0.95, 10],
 [0.7863247863247863, 'scale', 0.97, 12],
 [0.7863247863247863, 'scale', 0.93, 14],
 [0.7863247863247863, 'scale', 0.95, 14],
 [0.786019536019536, 'scale', 0.91, 14],
 [0.7857142857142857, 'scale', 0.91, 10],
 [0.7854090354090354, 'scale', 0.93, 12],
 [0.7854090354090354, 'scale', 0.95, 12],
 [0.7854090354090354, 'scale', 0.89, 14],
 [0.7851037851037851, 'scale', 0.99, 8],
 [0.7851037851037851, 'scale', 0.89, 10],
 [0.7851037851037851, 'scale', 0.91, 12],
 [0.7851037851037851, 'scale', 0.87, 14],
```

...

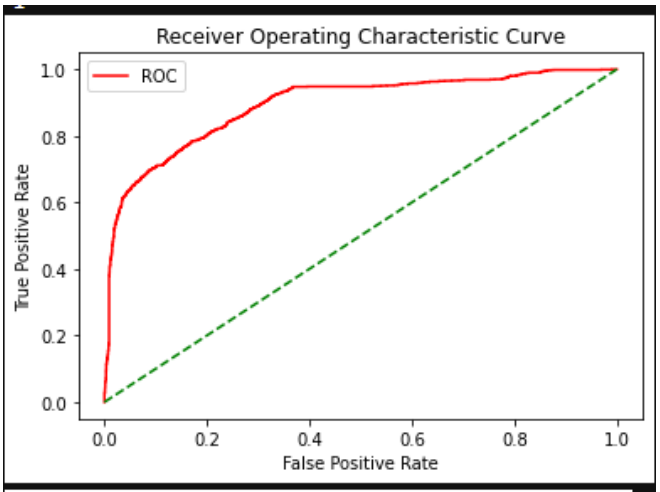
Com os melhores resultados (poly, degree=10), curvas ROC e matrizes de confusão

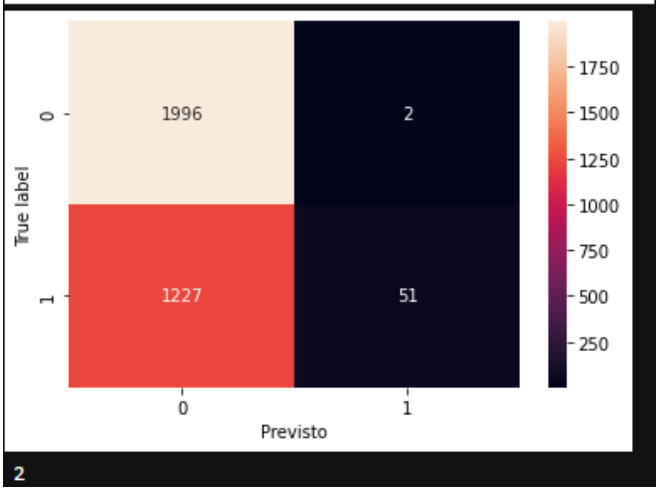
K-Fold	train_acc	val_acc	train_rec	val_rec	train_pre	val_pre	train_f1	val_f1
1	0.7893487109905021	0.6097560975609756	0.45858761987794244	0.22137404580152673	1.0	0.5272727272727272	0.6288105200239091	0.3118279569892473
2	0.7883310719131614	0.6432926829268293	0.4550218340611354	0.3233082706766917	1.0	0.6142857142857143	0.6254501800720288	0.4236453201970444
3	0.7883310719131614	0.5884146341463414	0.4629948364888124	0.1896551724137931	1.0	0.3492063492063492	0.6329411764705882	0.2458100558659218
4	0.7883310719131614	0.6524390243902439	0.4550218340611354	0.2857142857142857	1.0	0.6666666666666666	0.6254501800720288	0.4
5	0.7832428765264586	0.6402439024390244	0.45055889939810834	0.2	1.0	0.46938775510204084	0.6212211025489033	0.2804878048780488
6	0.7856173677069199	0.6310975609756098	0.44947735191637633	0.26153846153846155	1.0	0.576271186440678	0.6201923076923077	0.3597883597883598
7	0.7870464564259071	0.6269113149847095	0.45056867891513563	0.25925925925925924	1.0	0.6140350877192983	0.6212303980699638	0.3645833333333333
8	0.7900983384198034	0.5932721712538226	0.4551056338028169	0.21830985915492956	1.0	0.5849056603773585	0.6255293405928614	0.31794871794871793
9	0.7873855544252288	0.6636085626911316	0.45948275862068966	0.288135593220339	1.0	0.5666666666666667	0.6296515062020083	0.38202247191011235
10	0.7870464564259071	0.6422018348623854	0.45533391153512576	0.28	1.0	0.5645161290322581	0.6257449344457687	0.3743315508021391



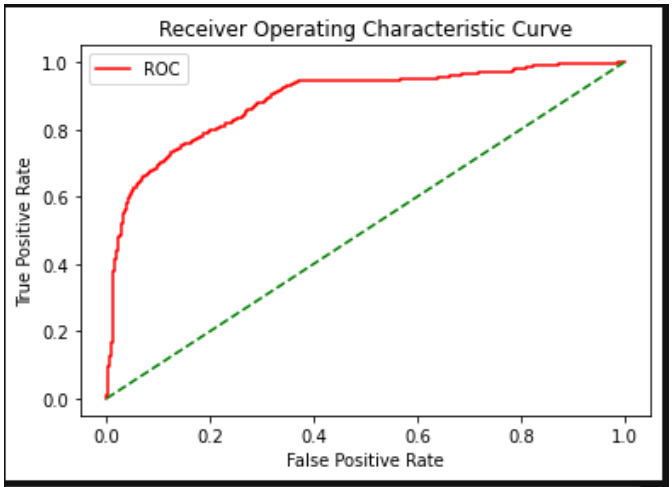


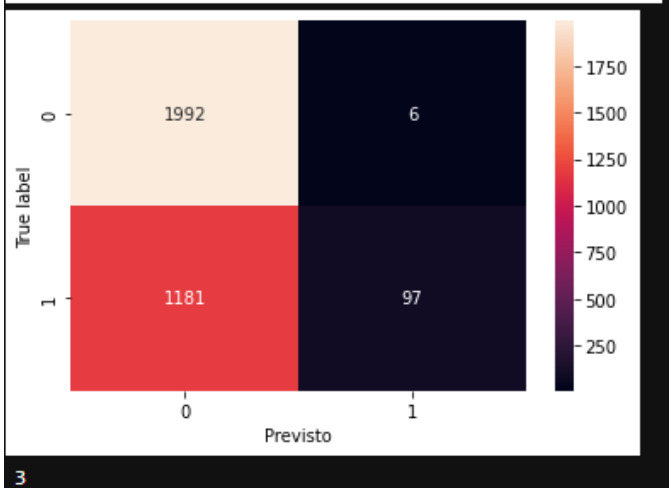
1



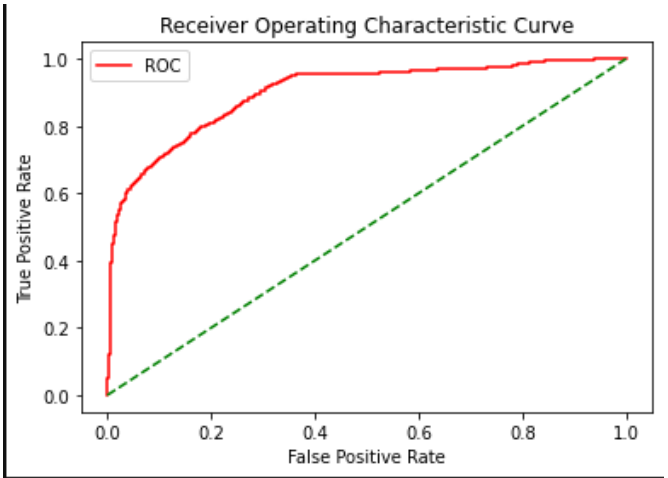


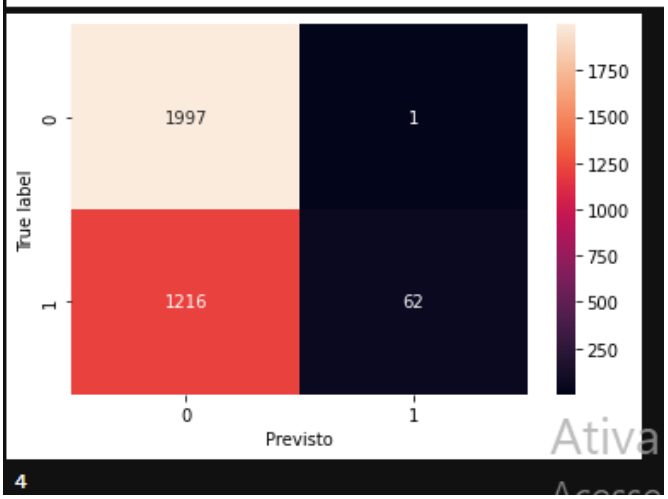
2


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

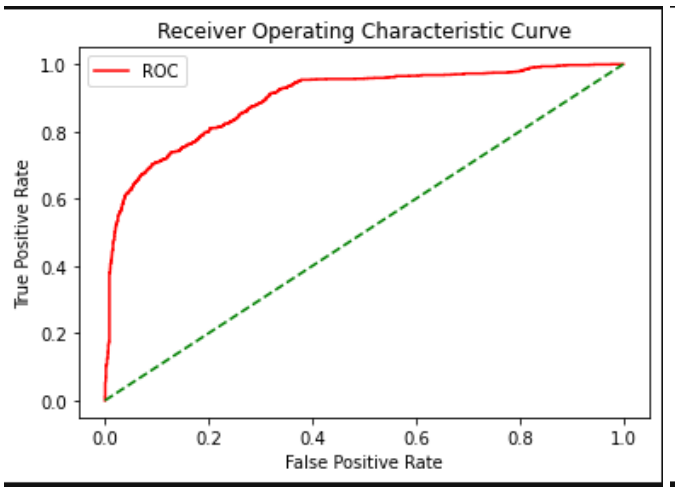

1992
6
1181
97
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

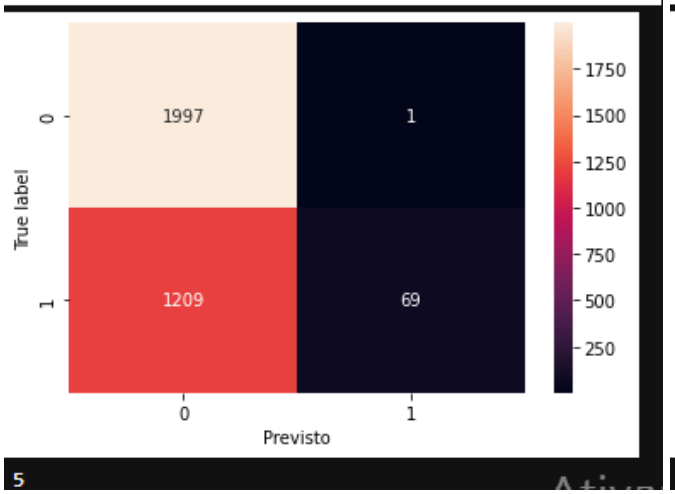
3


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

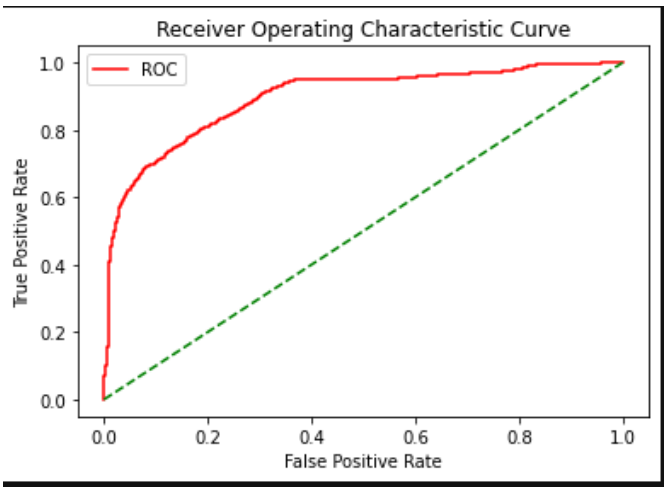

1997
1
1216
62
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

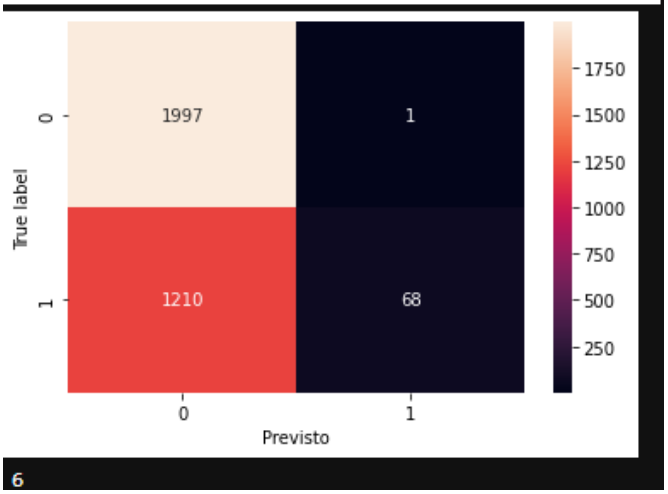
4


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

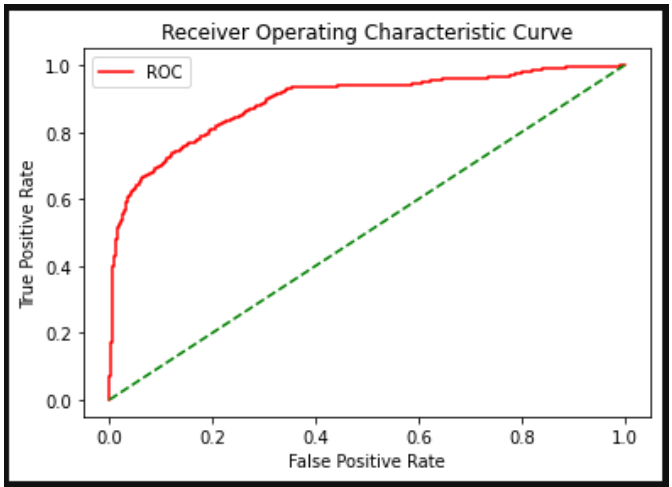

1997
1
1209
69
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

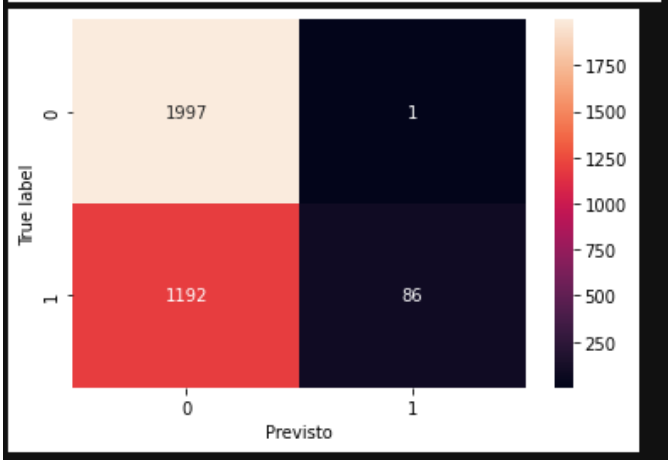
5


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

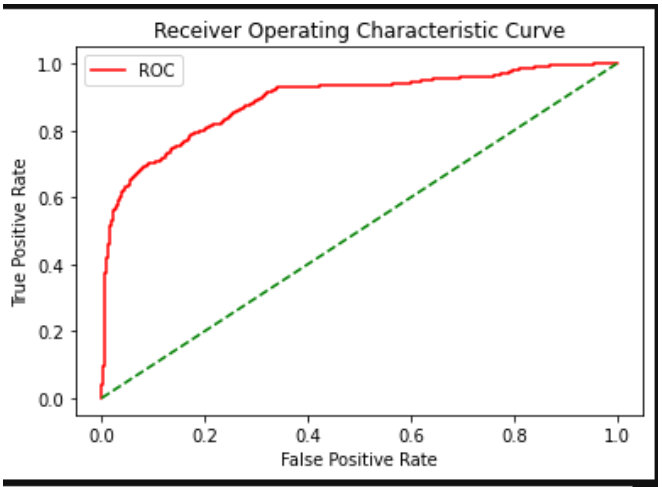

1997
1
1210
68
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

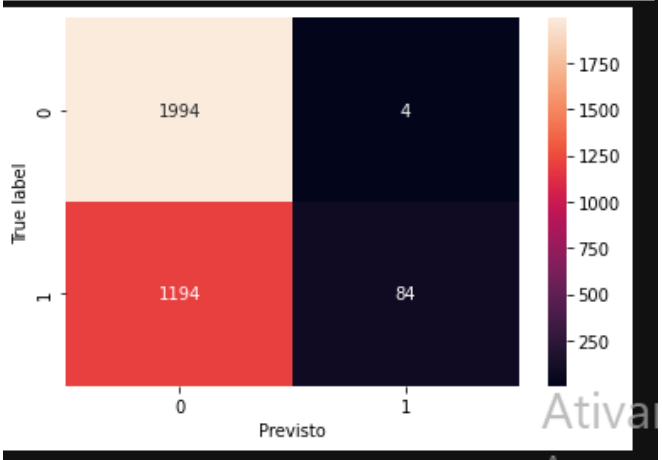
6


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

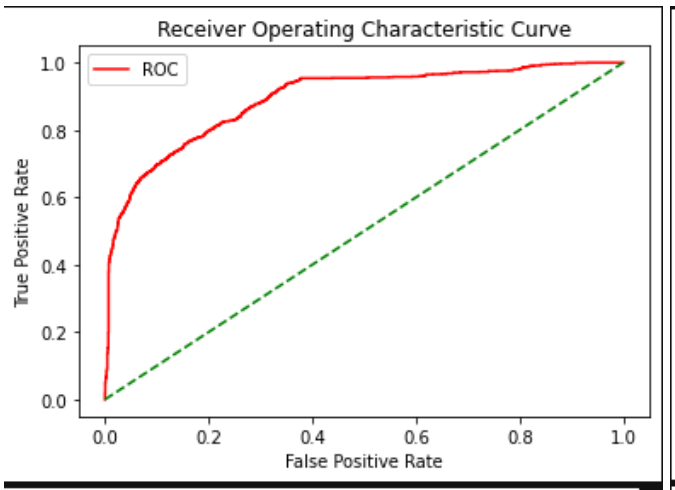

1997
1
1192
86
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

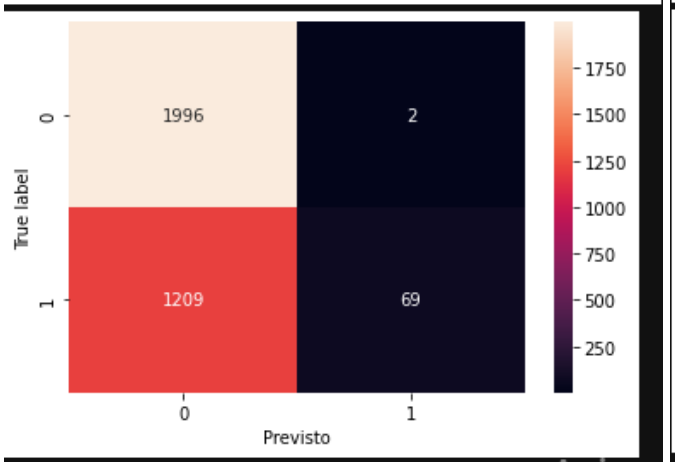
7


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

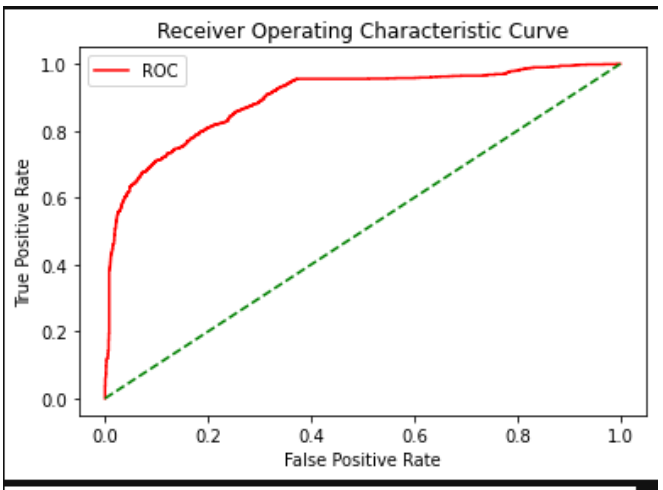

1994
4
1194
84
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

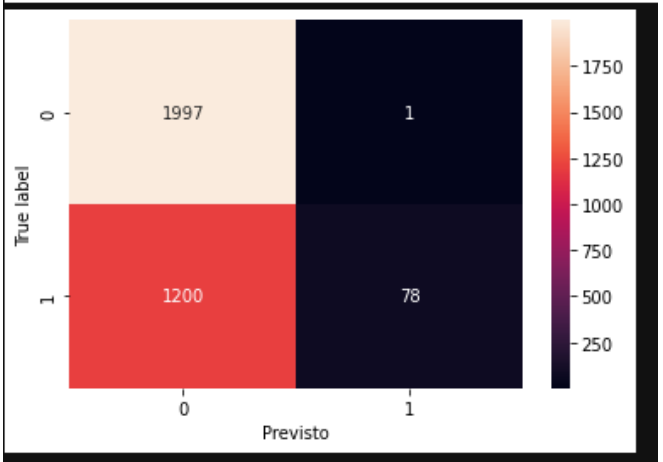
8


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


1996
2
1209
69
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

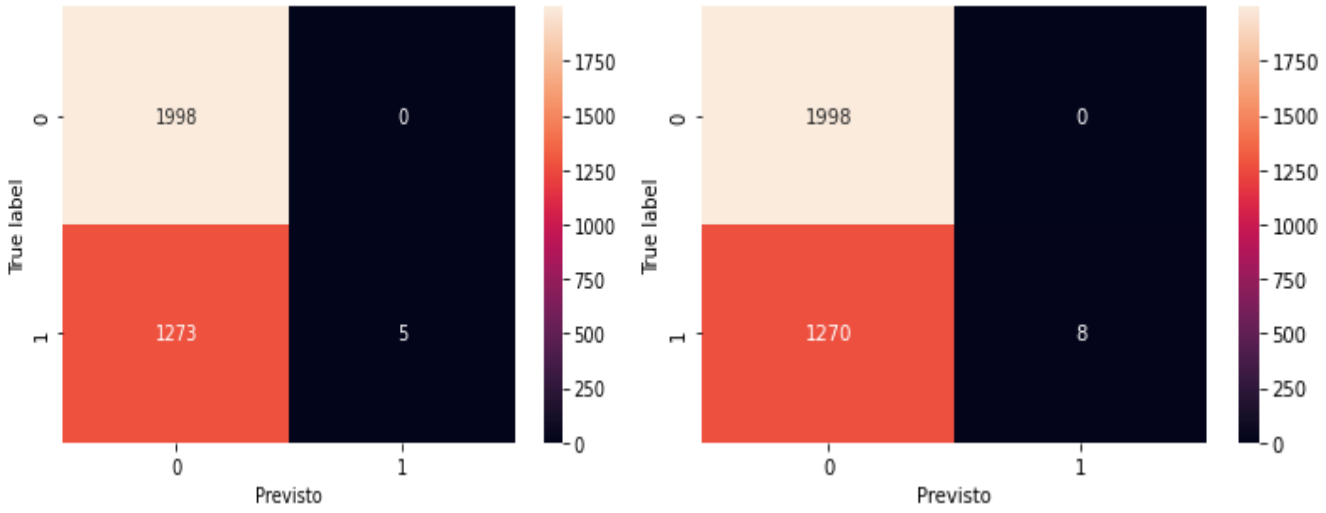
9


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

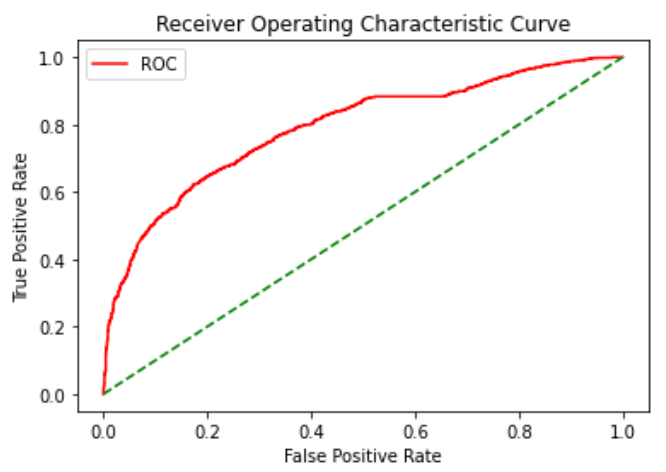

1997
1
1200
78
True label
Previsto
0
1
1750
1500
1250
1000
750
500
250

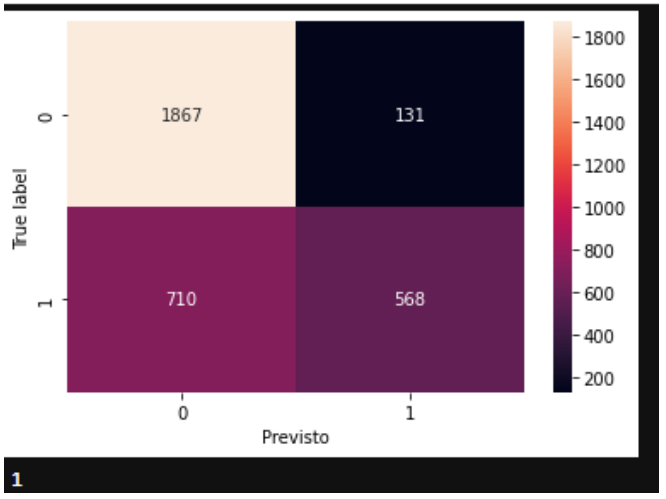
10

Uma tentativa com graus menores 3 e 4:

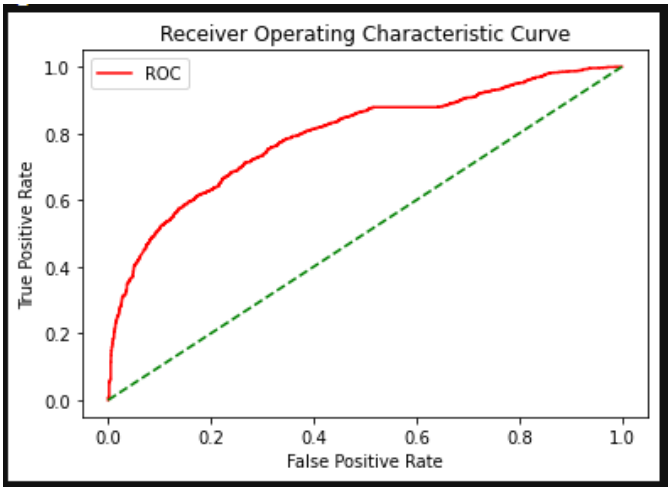


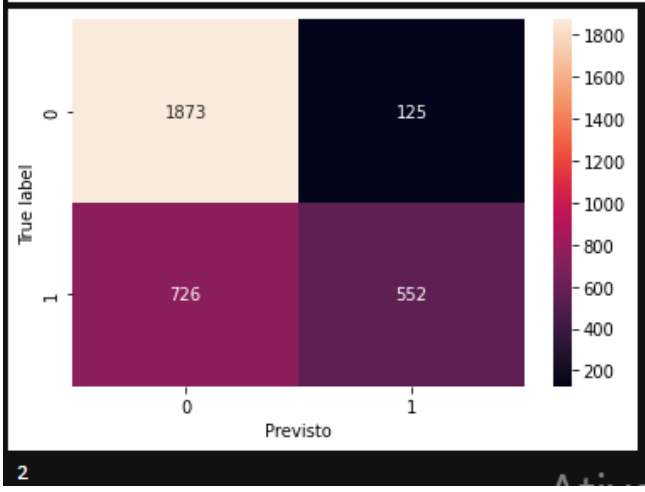
Com melhor para rbf, curvas ROC e matrizes de confusão:


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

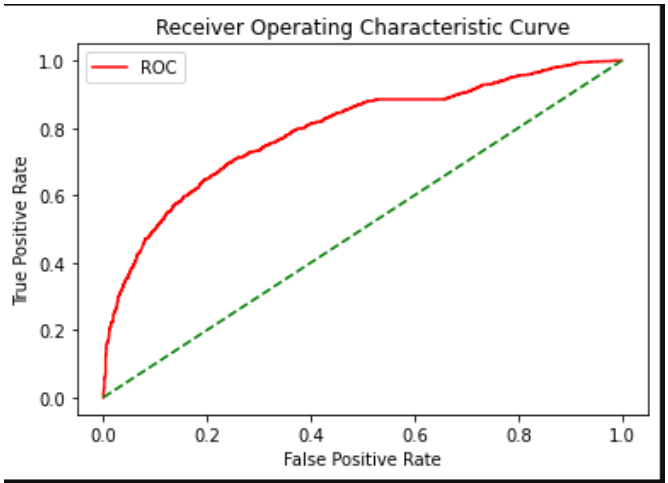

1867
131
710
568
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

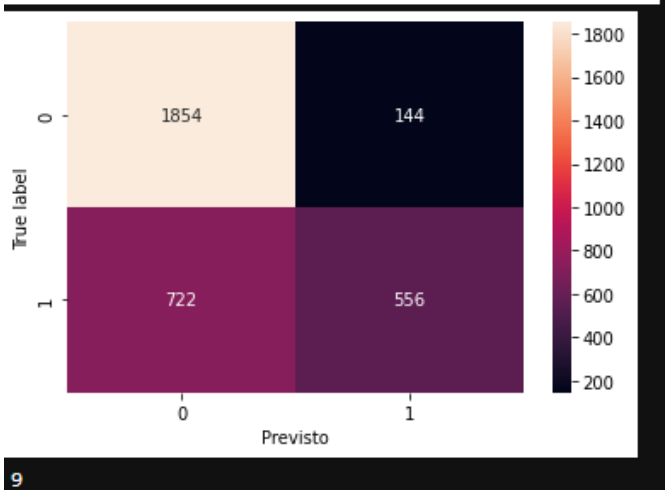
1


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

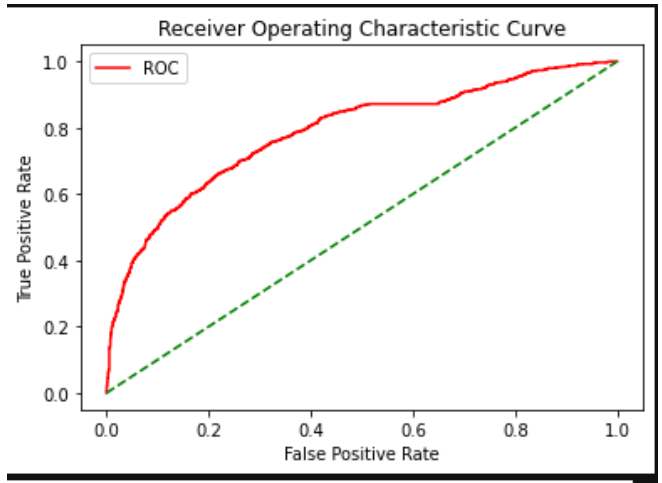

1873
125
726
552
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

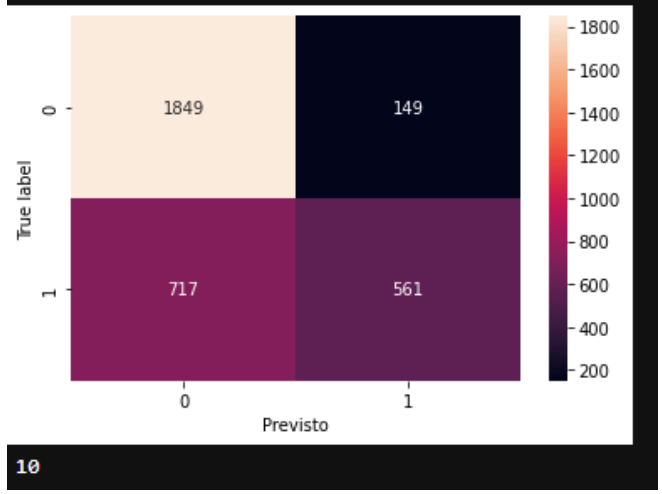
2


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


1854
144
722
556
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

9


Receiver Operating Characteristic Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


1849
149
717
561
True label
Previsto
0
1
1800
1600
1400
1200
1000
800
600
400
200

10

K-Fold	train_acc	val_acc	train_rec	val_rec	train_pre	val_pre	train_f1	val_f1
1	0.7218453188602443	0.7164634146341463	0.3567708333333333	0.373015873015873	0.8387755102040816	0.7704918032786885	0.5006090133982948	0.5026737967914439
2	0.7310040705563093	0.6554878048780488	0.3605263157894737	0.3115942028985507	0.8652631578947368	0.7049180327868853	0.5089783281733746	0.43216080402010054
3	0.7310040705563093	0.6829268292682927	0.37294015611448394	0.296	0.86	0.6981132075471698	0.5202661826981246	0.41573033707865165
4	0.7340569877883311	0.6676829268292683	0.3780487804878049	0.2846153846153846	0.8611111111111112	0.6981132075471698	0.5254237288135594	0.40437158469945356
5	0.7306648575305291	0.6615853658536586	0.3746747614917606	0.216	0.8554455445544554	0.675	0.5211097708082026	0.32727272727272727
6	0.7262550881953868	0.7134146341463414	0.3630174520069806	0.3409090909090909	0.8438133874239351	0.8653846153846154	0.5076266015863331	0.4891304347826087
7	0.7260088165479823	0.6972477064220184	0.365768896611642	0.33070866141732286	0.843687374749499	0.75	0.5103030303030303	0.4590163934426229
8	0.7321125805357749	0.6574923547400612	0.3748910200523104	0.2366412213740458	0.8548707753479126	0.7209302325581395	0.5212121212121212	0.3563218390804598
9	0.7263479145473042	0.6819571865443425	0.37618636755823986	0.23529411764705882	0.8384615384615385	0.6829268292682927	0.5193567599761764	0.3500000000000003
10	0.7348253645303493	0.6574923547400612	0.3842150910667823	0.256	0.8601941747572815	0.6274509803921569	0.5311750599520384	0.36363636363636365

4 Comparação entre os Modelos

Como as classes estão desbalanceadas tanto MLP quanto SVM apresentaram um "desequilíbrio" apenas ignorando a classe menos numerosa, com MLP foi possível controlá-lo com camadas de dropout gerando assim ótimos resultados como pode ser visto nas matrizes confusão. Já no caso do SVM os resultados iniciais já não foram tão bons quanto a MLP, mas o mais promissor parecia a função polinomial com altos graus (10, 11...) com acurácia abaixo, mais próxima da MLP, com uma análise mais profunda percebi que o modelo apenas ignorava a classe menos numerosa, tentei com outros parâmetros como com graus menores (o que apenas causou a redução da acurácia mais pouca ou nenhuma melhora nas outras métricas) e a utilização de outra função a RBF que reduziu a acurácia mais fez o modelo fazer predições mais equilibradas sendo assim o considerando o melhor parâmetro de função para SVM. Assim comparando com os resultados das outras técnicas da atividade anterior acredito que os melhores resultados ficam com MLP, Decision Tree e KNN, mais com a Decision Tree apresentando alto overfitting para tal e KNN mesmo com a maior acurácia apresentou alta dependência do conjunto dados apresentado, com apenas retirada de uma amostra dos dados para validação afetando bastante a performance do modelo logo chego a conclusão que o modelo com melhores resultados no geral foi o MLP.

Referências

[1] REDES NEURAIS ARTIFICIAIS PARA ENGENHARIA E CIÊNCIAS APLICADAS: AUTORES: IVAN NUNES DA SILVA, DANILO HERNANE SPATTI, ROGÉRIO ANDRADE FLAUZINO

[2] INTELIGÊNCIA ARTIFICIAL: AUTORES: Stuart J. Russell e Peter Norvig: SEÇÕES:18.7 e 18.9

[3] Problem formulations and solvers in linear SVM: a review : Vinod Kumar Chauhan · Kalpana Dahiya · Anuj Sharma