UNIVERSIDADE D
COIMBRA

Credit Card Default Prediction

Reconhecimento de Padrões

Mestrado em Engenharia Informática
2023/2024

11 de maio de 2024

Autores

João Moreira

joaomoreira@student.dei.uc.pt

PL2

2020230563

Conteúdo

1. Introdução

Este relatório tem como objetivo descrever o processo de classificadores simples que determinam se um individuo será capaz de pagar um credito bancário. Irei descrever processos como:

- Divisão de dados de treino e de teste
- Analise exploratória de dados
- Classificação de *features*
- Redução de *features*
- Criação de modelos de classificação

Fiz uso da linguagem de programação *python*, devido ao facto de ser muito utilizada pelos analistas de dados e à vasta variabilidade de modelos já previamente implementados e disponibilizados pelas *frameworks*:

- numpy
- pandas
- seaborn
- matplotlib
- scipy
- scikit-learn

Acabei por adotar o estilo tradicional dos analistas de dados, utilizei um *kernel de IPython* para auxiliar-me no desenvolvimento do código.

2. Data

Para este projeto, o *data set* usado encontra-se disponível em `https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients`. O primeiro passo foi fazer o *download* dos dados e converter o tipo de ficheiro .xls para .csv.

Antes de fazermos qualquer transformação ou visualização, vamos perceber o que cada coluna deste conjunto de dados representa. Podemos dividir as *features* em cinco grupos.

O primeiro grupo destina-se a informações sobre o utilizador:

- ID: id do cliente
- LIMIT_BAL: crédito inicial
- SEX: género do cliente (1: masculino, 2: feminino)
- EDUCATION: grau académico (1: graduação, 2: universidade, 3: ensino secundário, 4,5,6: desconhecido)
- MERRIAGE: estado civil (1: casado, 2:solteiro, 3: outro)
- AGE: idade do cliente

O segundo grupo, *PAY_** trata-se do histórico de pagamento dos meses: abril, maio, junho, julho, agosto, setembro de 2005, com uma escala de medida:

- -2: desconhecido, mas numa discussão pela *Internet* descobri que pode ser uma espécie de amortização
- -1: pagou devidamente
- 0: desconhecido
- 1: atraso de um mês
- 2: atraso de um mês
- 3: atraso de um mês
- ...
- 9: atraso de um mês

O terceiro grupo é o alor do extrato da conta dos meses de abril, maio, junho, julho, agosto e setembro.

O quarto grupo, *PAY_AMT_**, é o valor do pagamento anterior dos meses de abril, maio, junho, julho, agosto e setembro.

O último grupo é a classificação:

- 0: conseguiu pagar
- 1: não conseguiu pagar

Antes de passar à exploração de dados realizei uma filtragem para garantir que não existia valores em falta, *na values*. É um problema muito comum que temos de lidar na análise de dados. Felizmente, neste conjunto de dados isso não acontece.

2.1. Análise Exploratória de Dados

Com a análise exploratória de dados pretendo estudar e analisar a distribuição dos dados e com isso perceber se a aplicação de classificadores simples será uma tarefa fácil ou não.

2.1.1. Distribuição de Classes

A distribuição uniforme das classes é crucial em problemas de classificação, visto que os modelos de *machine learning* podem-se ajustar demasiado a uma determinada classe, devido ao desbalanceamento, e classificar a classe inversa incorretamente.

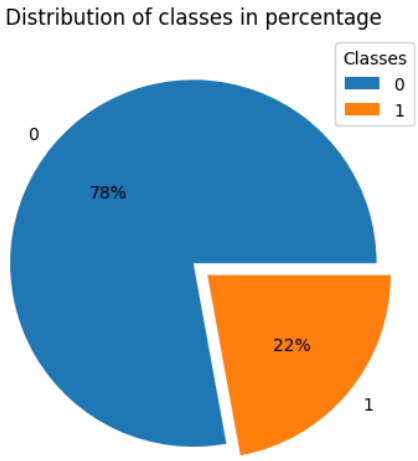


Figura 1: Distribuição das classes em percentagem

Como podemos ver pela figura 1, existe um tremendo desbalanceamento das classes, o que deve ser algo a ter em conta na forma como selecionamos os dados de treino, de forma a obter boas classificações.

2.1.2. Distribuição das Features

O que se pretende aqui é analisar visualmente a existência de boas *features*, ou seja, encontrar aquelas que descrevem corretamente a classe, e más *features*: aquelas que se sobrepões e não permitem uma boa descrição da classe.

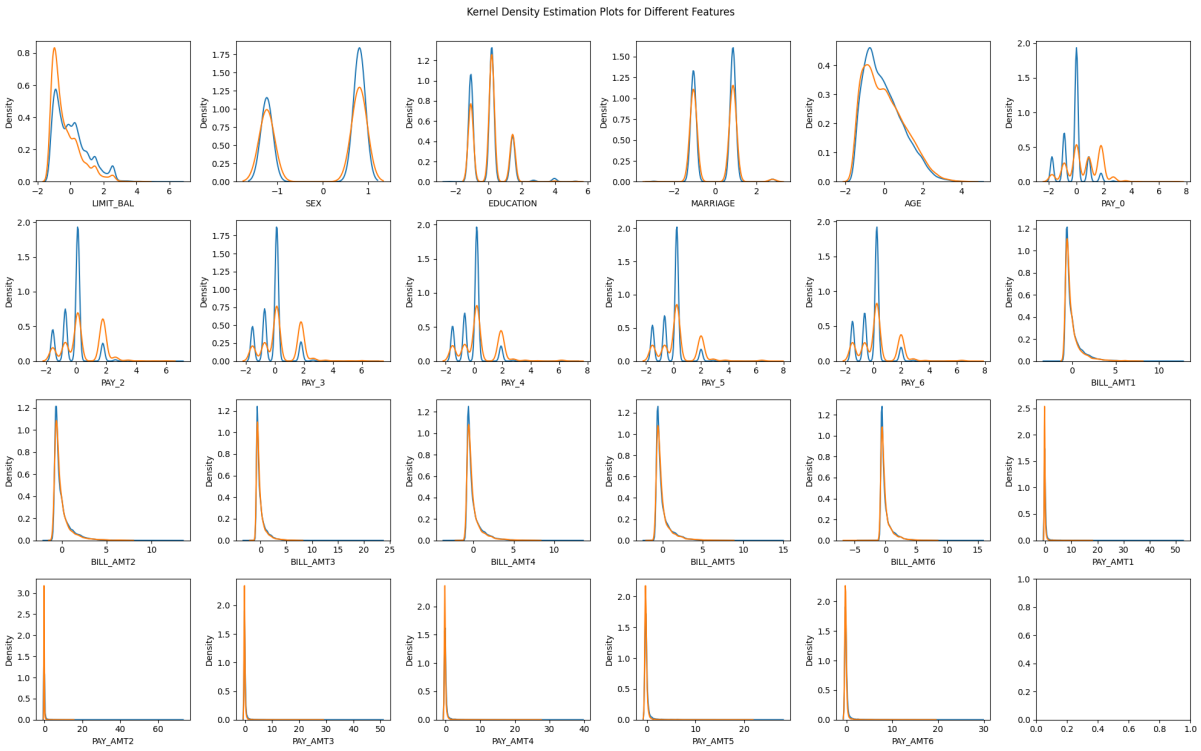


Figura 2: Distribuição das features por classe em formato KDE

Pela figura 2 a primeira coisa que podemos observar é que nenhuma das nossas *features* aparenta seguir uma distribuição normal e que parte delas são variáveis categoricas e não numé-

ricas.

Outra coisa que podemos ver que qualquer das *features* das linhas 2 e 3 são péssimas, para além de seguirem a mesma distribuição estão nas mesmas proporções, ou seja, usa-las para classificação nunca teremos uma classificação superior a 50%. As *features* que fogem mais desse padrão são as do grupo *PAY_**.

As restantes, seguem a mesma distribuição, mas por vezes em proporções diferente.

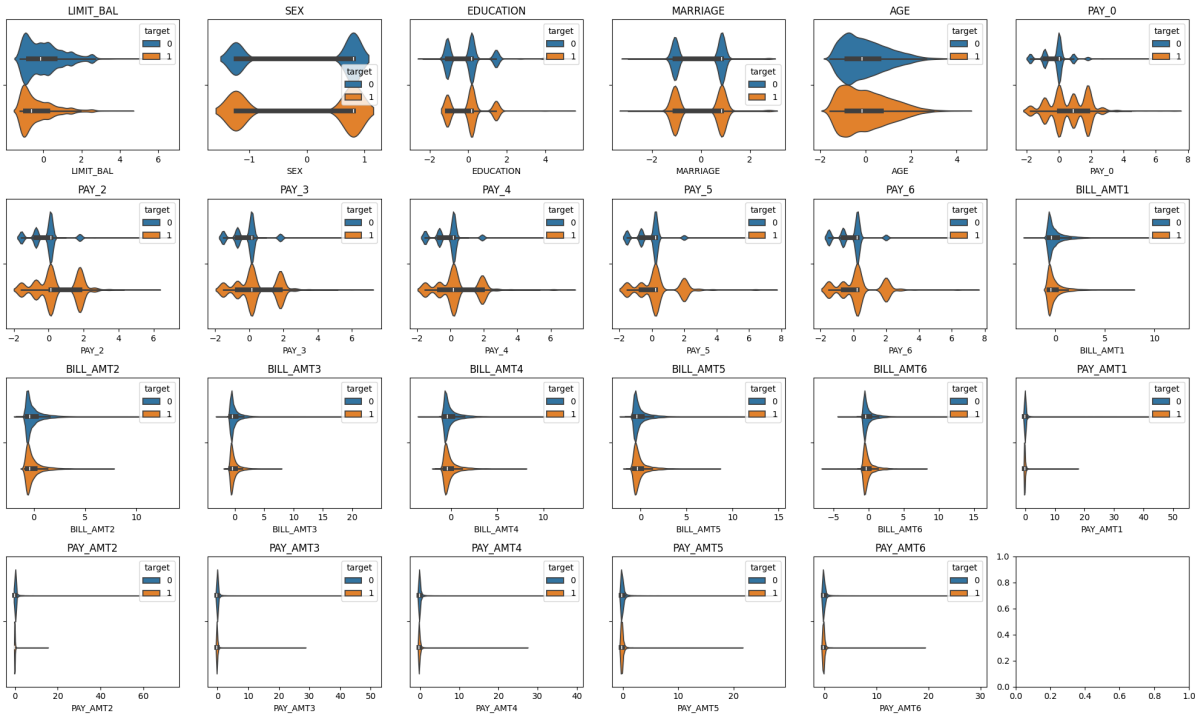


Figura 3: Distribuição das features por classe em formato de violino

Ao analisar as *features* do grupo *PAY_** na figura 3, vemos que se o cliente tem um mês ou dois em atraso é um forte indicativo que não conseguirá pagar o crédito.

2.1.3. Correlação de Dados

Outra forma gráfica muito boa de se analisar é a matriz de correlação. Ela permite identificar quais *features* estão fortemente correlacionadas entre si e como elas se relacionam com a variável alvo.

Ao olhar para a matriz de correlação devemos procurar por pares de variáveis fortemente correlacionadas, pois significa que elas são redundantes e podem ser eliminadas. Devemos também, procurar por variáveis fortemente correlacionadas com a variável *target*.

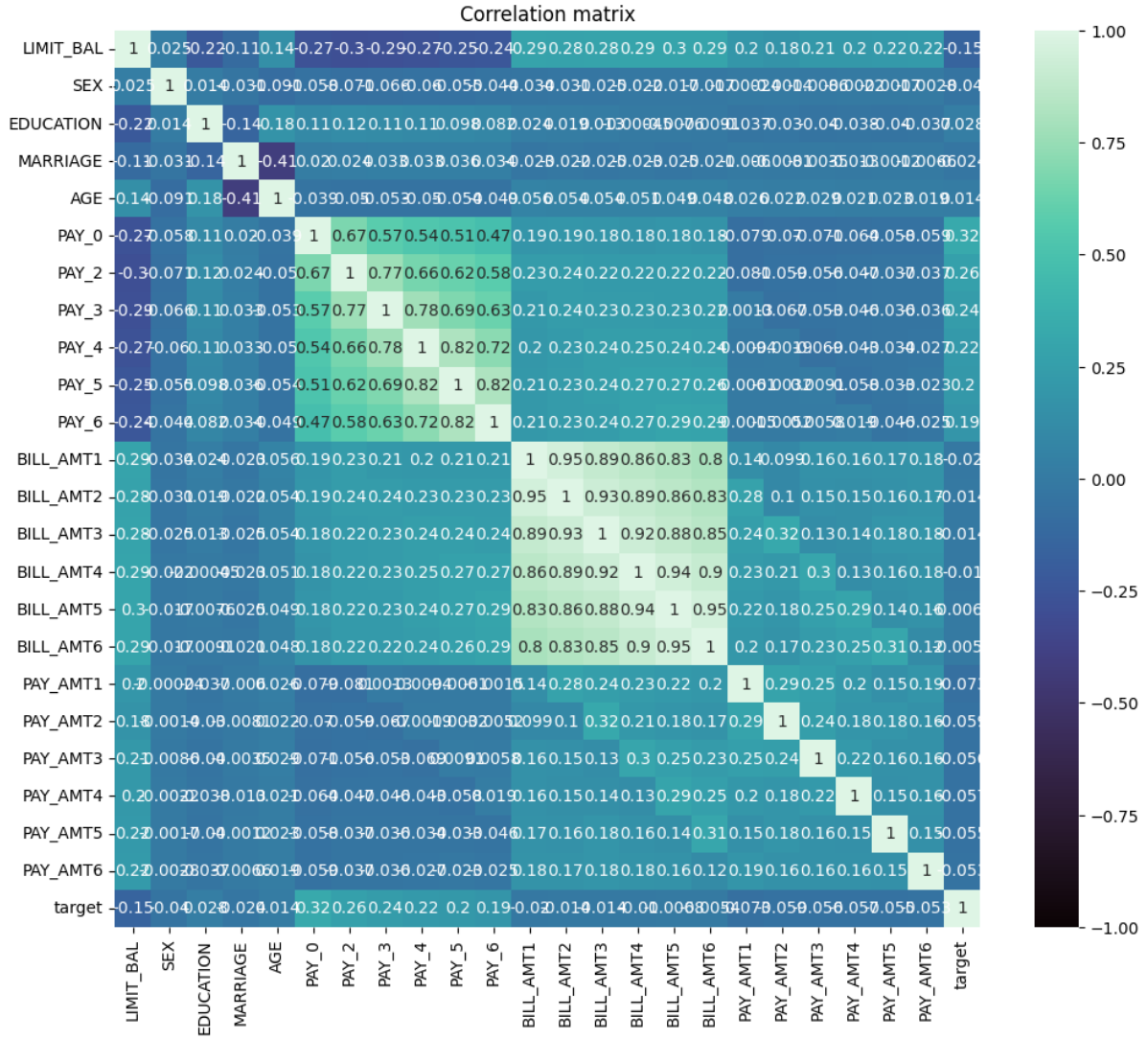


Figura 4: Correlação dos dados

Pela figura 4 conseguimos ver que o grupo *BILL_AMT_** está fortemente correlacionado entre si, ou seja, seria boa ideia reduzir ou codificar todo aquele bloco numa só variável, de forma a eliminar a multicolinearidade.

Da mesma forma no grupo *PAY_** está muito correlacionado entre ele. No entanto, ao mesmo tempo está um pouco correlacionado com o *target*. Reduzir este grupo exige muito cuidado pois poderemos eliminar informações necessária para uma boa classificação.

3. Seleção e Redução de Features

Na secção anterior tentamos tentamos supor quais seriam as melhores *features* para uma boa classificação e vimos que não era uma tarefa muito simples. Aqui faremos o mesmo com recurso a algumas técnicas de seleção e redução de *features* mais robustas.

3.1. Teste Kolmogorov–Smirnov

O teste de Kolmogorov–Smirnov quantifica a diferença entre a função de distribuição cumulativa empírica de uma amostra de dados e a função de distribuição cumulativa teórica de uma

distribuição padrão normal.

Sendo assim, podemos usar este teste para ver se os nossos dados seguem uma distribuição normal.

3.2. Teste Kruskal-Wallis

Nem sempre os dados seguem distribuições normais. Quando isso acontece, resta-nos usar testes não-paramétricos, como no caso do Kruskal-Wallis. Estes teste determina o poder discriminativo das *features* por classificação

$$H = \frac{12}{n(n+1)} \sum_{i=1}^{C} n_i (R_i - \overline{R})^2$$

Desta forma, podemos determinar quais são as *features* com mais impacto na classificação.

Apesar de este método dizer-nos qual é a melhor *feature*, ele olha *feature* a *feature* individualmente e não em conjunto, o que provoca muita redundância, visto que elas podem ser muito iguais e poderiam ser eliminadas.

3.3. ROC Curve

3.4. Principal Component Analysis

A PCA é um procedimento estatístico que usa a transformação ortogonal para projetar um conjunto de variáveis correlacionadas num conjunto não correlacionado de variáveis.

Esta é uma tecnica de *machine learning* não supervisionada, ou seja, apenas com ajuda da matriz de co-variância calcula a distribuição dos dados e não toma qualquer informação das classes.

O seu principal objetivo é reduzir a dimensional do *data set* e ao mesmo tempo preservar a maior quantidade de informação possível, sem qualquer informação sobre a variável alvo.

Podemos calcular a percentagem de variância preservada pela expressão

$$R(\%) = \frac{\sum_{i=1}^{D} \lambda_i^2}{\sum_{i=1}^{M} \lambda_i^2} \times 100$$

3.5. Linear Discriminant Analysis

LDA é uma técnica de redução de *features*. O objetivo deste método é projetar os dados em direções que maximiza a distância entre as medias das classes e que minimiza a variância entre cada classe.

Sendo assim, o treino deste modelo é definir um critério para maximizar que tenha em conta a separabilidade e compactação de classes

$$J(W) = \frac{|W^t s_B W|}{|W^t s_w W|}$$

4. Classificação de Dados

Depois de termos feito toda esta analise de dados e de *features*, chega o momento de classificar os dados, sendo assim, alguns modelos foram construídos.

4.1. Minimum Distance Classifier

Este é um classificador super simples, a cada classe atribui uma classe protótipo, cuja é a média dos objetos da classe.

Como o próprio nome indica, o classificador irá prever um objeto com a *label* da classe mais próxima. A distância entre o objeto e o centroide pode ser com qualquer tipo de métrica de distância, neste caso usaremos a distância Euclidiana e a distância de Mahalanobis.

Sendo assim, para a distância Euclidiana estaremos a resolver o problema

$$x \in \omega_k \; if \; \max_k \{m_k^t x - 0.5 m_k^t m_k\}$$

e para a distância de Mahalanobis

$$x \in \omega_k \; if \; \max_k \{m_k^t C^{-1} x - 0.5 m_k^t C^{-1} m_k\}$$

4.2. Fisher LDA Classifier

Este classificador não tem nada de novo para além do que já falamos até agora, simplesmente usamos a LDA para reduzir as variáveis e aplicamos a MDC.

4.3. Naive Bayes Classifier

O Classificador *Naive Bayes* é um modelo estatístico baseado na teoria de *Bayes*, que é utilizado em problemas de classificação.

Durante a fase de previsão, o modelo usa as distribuições gaussiana calculadas para calcular a probabilidade de uma nova instância pertencer a cada classe, minimizando o risco/custo total esperado.

4.4. k-Nearest Neighbors Classifier

O *k-Nearest Neighbors* procura os k vizinhos mais próximos e classifica uma nova amostra de acordo com a classe de maior prevalência.

Durante a construção deste classificador o desafio é encontrar o melhor k que maximiza a performance do algoritmo.

Apesar de o algoritmo ser simples de perceber teoricamente e até de implementar, pode ser que se torne um algoritmo exaustivo para grandes volumes de dados, devido à procura dos pontos mais próximos. Sendo assim, deixo uma referência para uma estrutura de dados, *K-D Tree*, que ajuda-nos a calcular os pontos mais próximos de forma eficiente, `https://en.wikipedia.org/wiki/K-d_tree`.

4.5. Support Vector Machine Classifier

A *SVM* é um classificador muito poderoso, baseado na procura de um hiper-plano que maximiza a separação entre as classes e minimiza os erros de classificação em treino.

Para minimizar o erro em treino, a *SVM* tenta que a margem entre os pontos mais próximo das duas classes deve ser o máximo possível.

4.6. Random Florest Classifier

No *Random Florest*, múltiplas árvores de decisão são criadas usando diferentes subconjuntos aleatórios de *features*. Partimos da crença que cada árvore de decisão torna-se especialista num determinado conjunto de características.

Quando queremos classificar uma nova amostra, passamos-a por todas as árvores e aceitamos o resultado mais popular.

5. Resultados

5.1. Redução de Features

5.1.1. Teste K-S

Para todas as *features* executei o teste com um grau de confiança de 95%, pelo que rejeitaremos a hipótese nula para um valor de p menor que 0.05:

- H_0: A amostra segue uma distribuição normal
- H_1: A amostra não segue uma distribuição normal

e obtive os seguintes *p-values*.

Feature	P-Value	Feature	P-Value
'LIMIT_BAL'	0.0	'BILL_AMT1'	0.0
'SEX'	0.0	'BILL_AMT2'	0.0
'EDUCATION'	0.0	'BILL_AMT3'	0.0
'MARRIAGE'	0.0	'BILL_AMT4'	0.0
'AGE'	$3.962e-233$	'BILL_AMT5'	0.0
'PAY_0'	0.0	'BILL_AMT6'	0.0
'PAY_2'	0.0	'PAY_AMT1'	0.0
'PAY_3'	0.0	'PAY_AMT2'	0.0
'PAY_4'	0.0	'PAY_AMT3'	0.0
'PAY_5'	0.0	'PAY_AMT4'	0.0
'PAY_6'	0.0	'PAY_AMT5'	0.0
'PAY_AMT6'	0.0		

Tabela 1: P-Values obtidos para o K-S teste

Sendo assim, para qualquer que seja a *feature* somos obrigados a rejeitar a hipótese nula e assumir que os dados não seguem uma distribuição normal.

5.1.2. Teste de Kruskal-Wallis

Na tabela 2 estão os *rankings* classificados pelo *K-W teste*.

Feature	P-Value	Feature	P-Value
PAY_AMT6	23702.21	LIMIT_BAL	3102.06
PAY_AMT3	21591.27	AGE	2694.41
PAY_AMT2	20982.74	PAY_0	1331.28
PAY_AMT4	19374.24	MARRIAGE	1108.18
PAY_AMT1	19270.67	PAY_2	615.24
PAY_AMT5	19210.45	PAY_3	482.09
BILL_AMT6	10893.89	PAY_4	382.74
BILL_AMT5	10809.71	EDUCATION	365.70
BILL_AMT1	10608.73	PAY_5	297.08
BILL_AMT4	10216.06	SEX	176.91
BILL_AMT2	9874.18	PAY_6	126.67
BILL_AMT3	9671.52		

Tabela 2: Ranking obtidos para o K-W teste

Como podemos ver pela tabela 2, o *K-W teste* classificou como melhores *features* aquelas que previámos ser as piores e vice-versa.

Iremos testar o primeiro grupo, aquele que fica acima do *threshould* desenhado na figura 5, ou seja, as *features*: PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5 e PAY_AMT6.

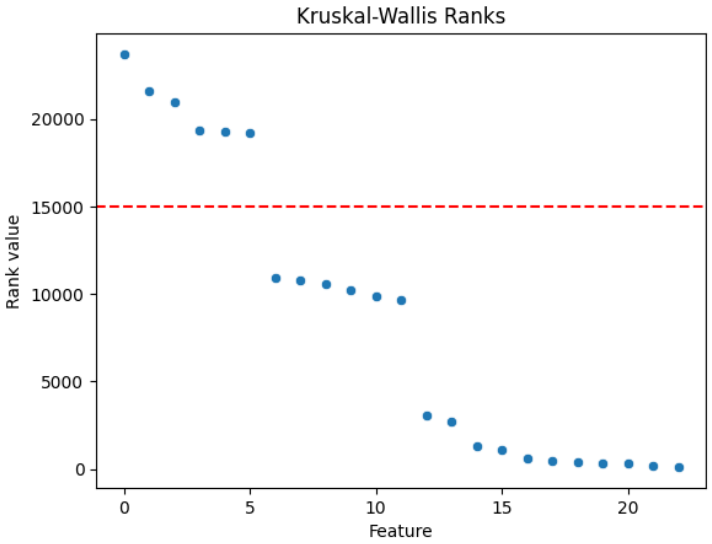


Figura 5: Visualização dos ranking obtidos para o K-W teste

5.1.3. ROC

Na figura6 estão presentes os resultados da *ROC* e a respetiva *AUC* para todas as features do *dataset*.

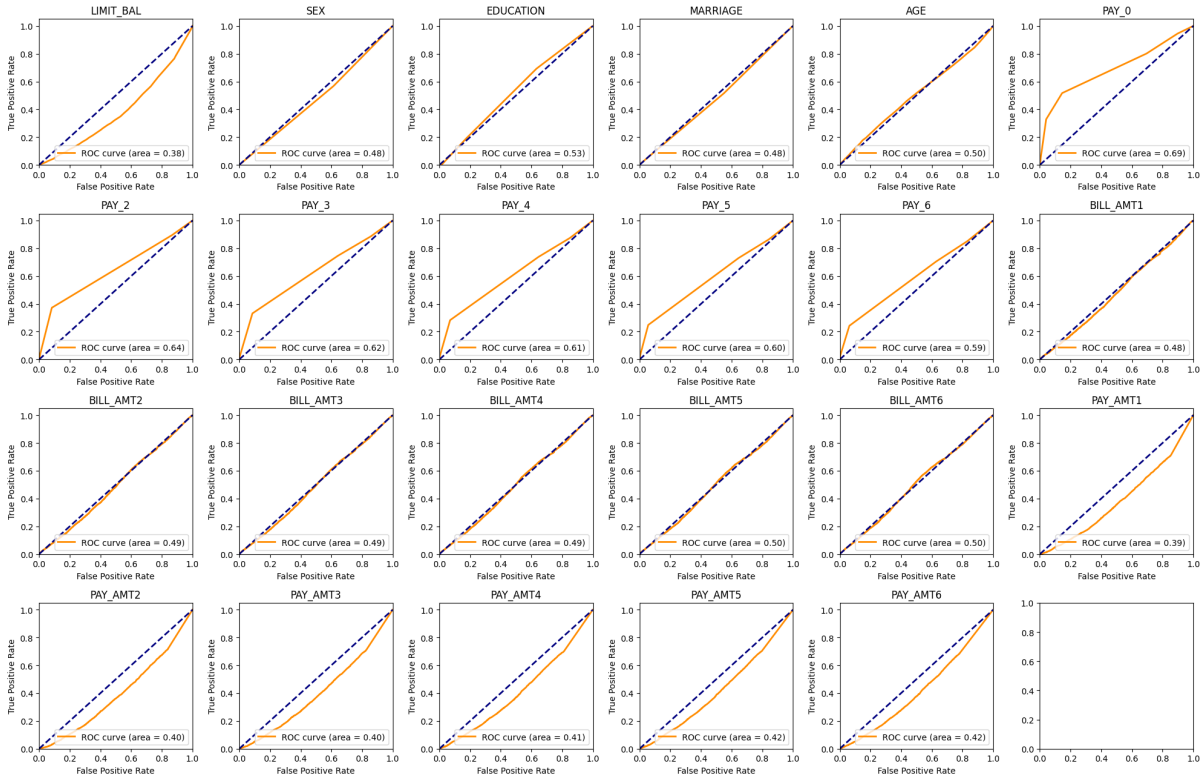


Figura 6: Visualização das curvas ROC para todas as features

As *features PAY_** são as únicas com a *AUC* acima de 60%. Todas as outras aprensentam a *AUC* abaixo dos 50%. Esta técnica, valida a nossa análise exploratória quando levantamos a hipótese de que estas seriam as melhores *features* para a classificação.

5.1.4. PCA

Em ordem a escolher os melhores vetores a projetar os dados, executei o método de *PCA*. Na figura 7 é possível ver os resultados obtidos.

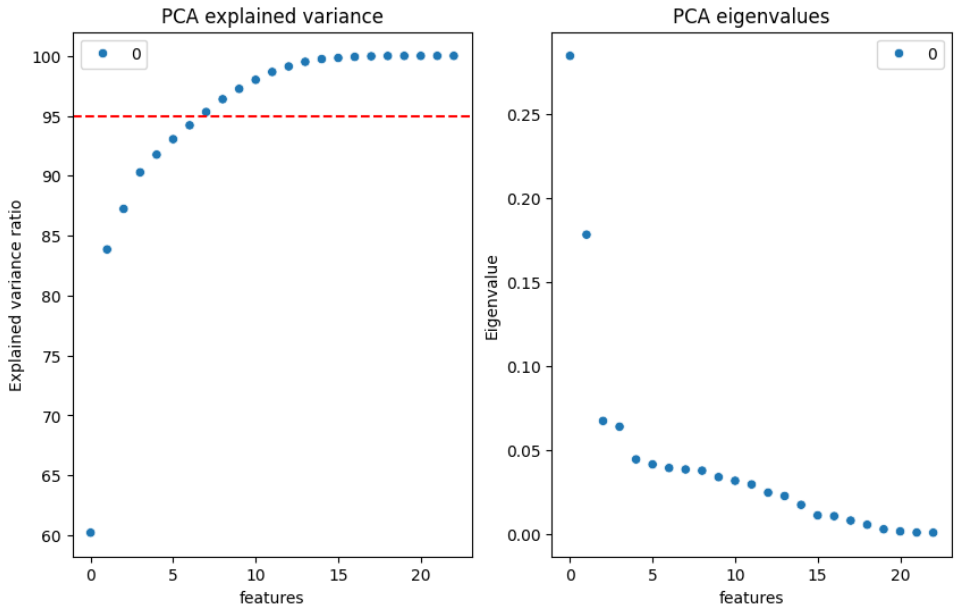


Figura 7: Visualização da variância preservada e dos vetores próprios para a PCA

Assim sendo, escolhi usar as variáveis que me garantem pelo menos a preservação de 95% da informação, ou seja, as *features*: 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0', 'PAY_2' e 'PAY_3'.

Visto que é impossível a visualização dos dados num plano de oito dimensões, na figura 8 está a distribuição das novas variáveis projetadas pela *PCA*.

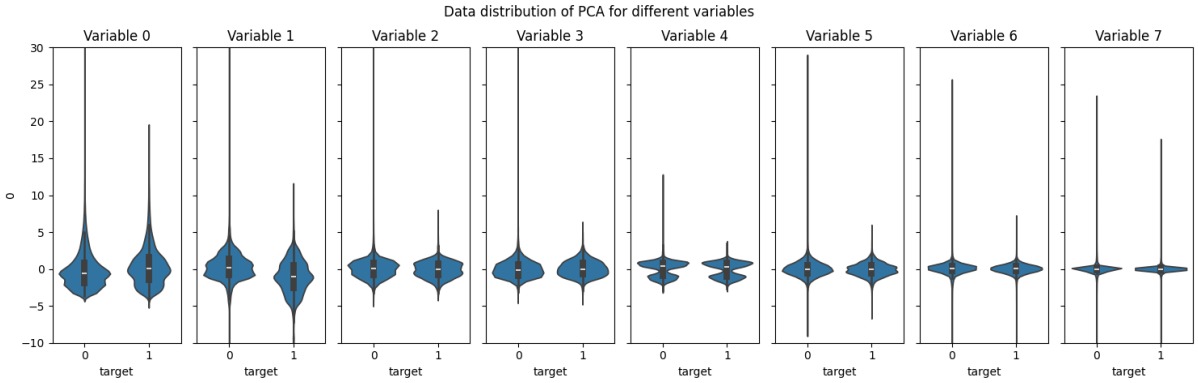


Figura 8: Visualização das variáveis projetadas pela PCA

Como era de imaginar, continua a ser difícil de encontrar uma solução para separar os dados linearmente.

5.1.5. LDA

Uma vez que temos apenas duas classes, com a *LDA* somos obrigados a projetar os dados a uma dimensão. Apesar de a LDA ser uma ferramenta de redução de *features*, a redução da dimensão é muito grande, o que acaba por resultar numa projeção difícil de separar, como é visível na figura 9.

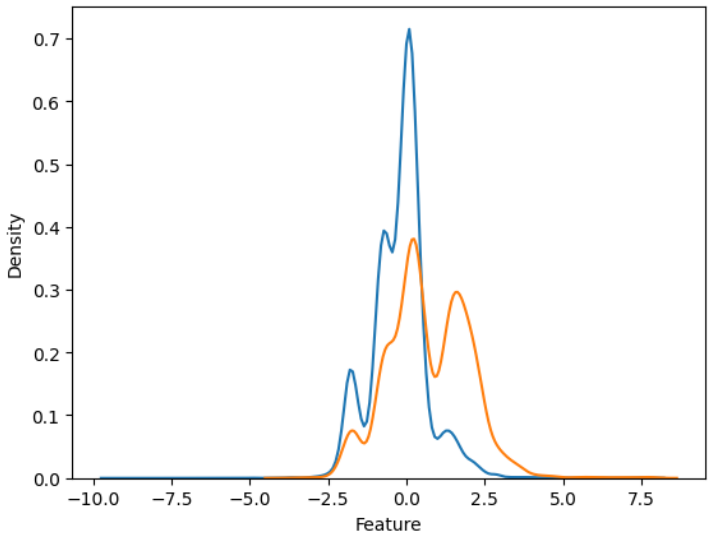
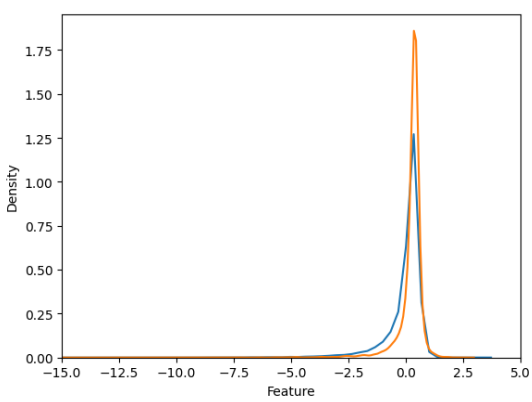


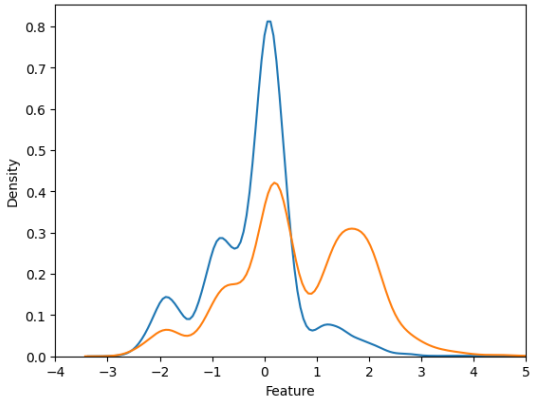
Figura 9: Visualização da projeção pela LDA

Pela razão anterior, tentei aplicar a *LDA* com diferentes *features*. Na figura 10b selecionei as variáveis: 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6'. Estas apresentam os mesmos resultados face não selecionar nada.

Na figura 10a passei à *LDA* variáveis que obtiveram melhores resultados no *k-s test*. Estas obtiveram um resultado péssimo.



(a) LDA com as features classificadas pelo k-s teste



(b) LDA com as features escolhidas visualmente

Figura 10: Distribuição da LDA face a diferentes features escolhidas

5.2. Classificação

Nesta secção, apresentarei e analisarei os resultados obtidos pelos classificadores, para além de relatar algumas estratégias empregadas com o objetivo de alcançar melhor desempenho.

Para assegurar uma avaliação justa, todos os classificadores foram executados 30 vezes. Em cada execução, o conjunto de dados foi dividido aleatoriamente em 80% para treino e 20% para teste, com reposição. A aleatoriedade foi controlada com a utilização de uma semente fixa em cada execução, garantindo imparcialidade nesse aspeto.

É importante ressaltar que os conjuntos de dados de treino e teste foram normalizados separadamente, visando uma análise mais precisa.

Devido à natureza de aleatoriedade do classificador de Floresta Aleatória, também foi definida uma semente para a geração das árvores.

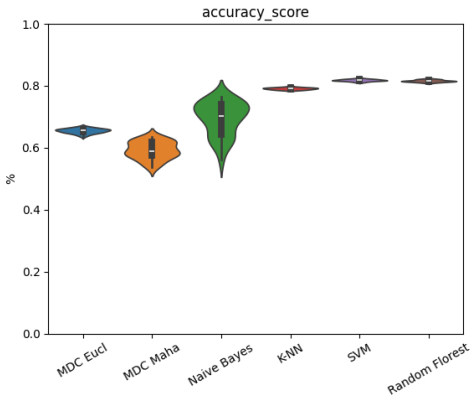
Todos os resultados apresentados serão referentes aos conjuntos de teste e não de treino.

5.2.1. Classificação sem seleção de features

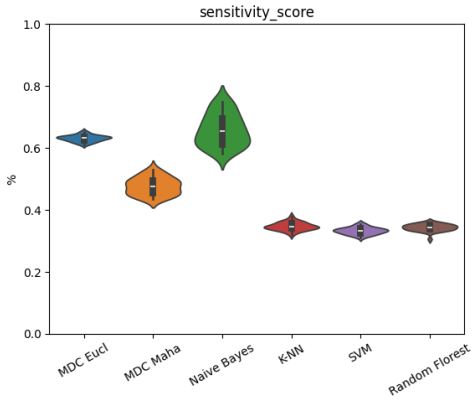
Antes de aplicar qualquer tipo de seleção de *features* aos classificadores, testei os classificadores sem qualquer transformação além da normalização.

Este passo é crucial para compreender quais os classificadores que se adequam melhor ou pior aos dados. Além disso, permite-nos comparar os resultados com classificadores que iremos desenvolver no futuro.

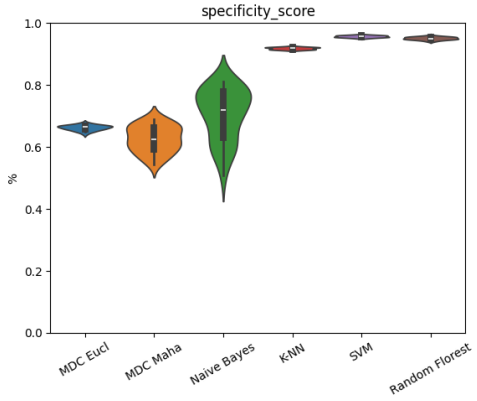
Na figura 11 é exibido os resultados obtidos.



(a) Exatidão dos classificadores com dados crus



(b) Sensibilidade dos classificadores com dados crus

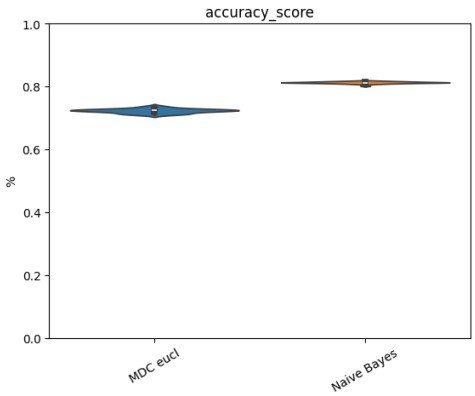


(c) Especificidade dos classificadores com dados crus

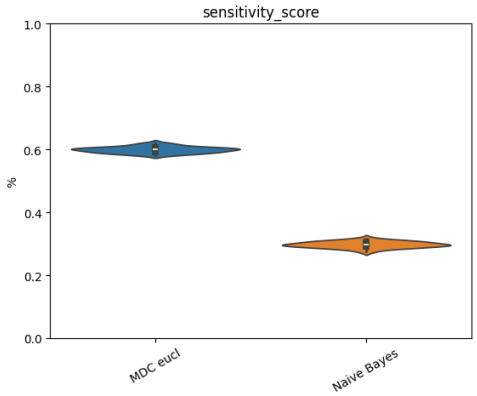
Figura 11: Métricas de performance dos classificadores com dados crus

5.2.2. Classificação com LDA

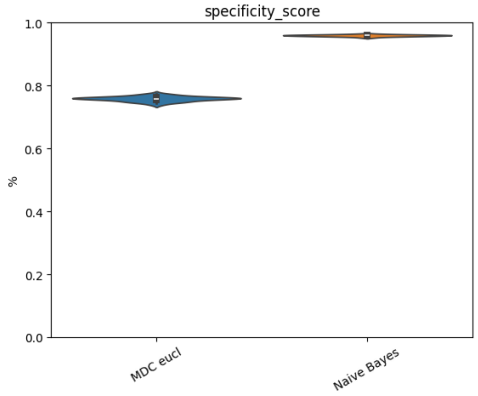
Na figura 12 podemos ver os resultados obtidos para a classificação dos dados com os classificadores: *Minimum Distance Classifier* com a distância euclidiana e do *Naive Bayes Classifier*.



(a) Exatidão dos classificadores com as features da LDA



(b) Sensibilidade dos classificadores com as features da LDA



(c) Especificidade dos classificadores com as features da LDA

Figura 12: Métricas de performance dos classificadores com as features da LDA

Escolhi especificamente esses dois classificadores porque faziam mais sentido de aplicar à *LDA*, porque Neste caso a redução resultou em um conjunto de dados uni-dimensional.

Sendo assim, não faz sentido aplicar os classificadores:

- *MDC* com distância de *Mahalanobis*, porque não existe uma direção de variação.
- *SVM* porque é comum aplicar a conjuntos de dados de alta dimensão.
- *K-NN* porque a uma dimensão as amostras ficam muito sobrepostas e podemos perder a noção de distância.
- *Random Florest* porque precisávamos de varias *features* para a construção das árvores.

Ao analisar os gráficos 12a) e 12b), observamos que o Classificador de Distância Mínima consegue manter uma boa exatidão sem perder precisão, enquanto o *Naive Bayes* tem uma sensibilidade próxima da distribuição da classe minoritária.

Comparando estes resultados com os resultados da classificação com dados cruz, o *Naive Bayes*, apesar de apresentar uma sensibilidade muito pior, conseguiu manter um baixo desvio padrão, o que é positivo. Por outro lado, o Classificador de Distância Mínima perdeu muita pouca sensibilidade, e ganhou um pouco de exatidão e especificidade. Estou satisfeito com esse resultado, pois conseguimos reduzir as *features* para uma dimensão e manter resultados semelhantes com um classificador relativamente simples.

5.2.3. Classificação com PCA

De forma a estimar os melhores valores para k do classificador *K-k-Nearest Neighbors*, executei o algoritmo trinta vezes para cada valor de $k \in [1, 30[$ e calculei a média dos resultados.

Na figura 13 apresento os resultados obtidos

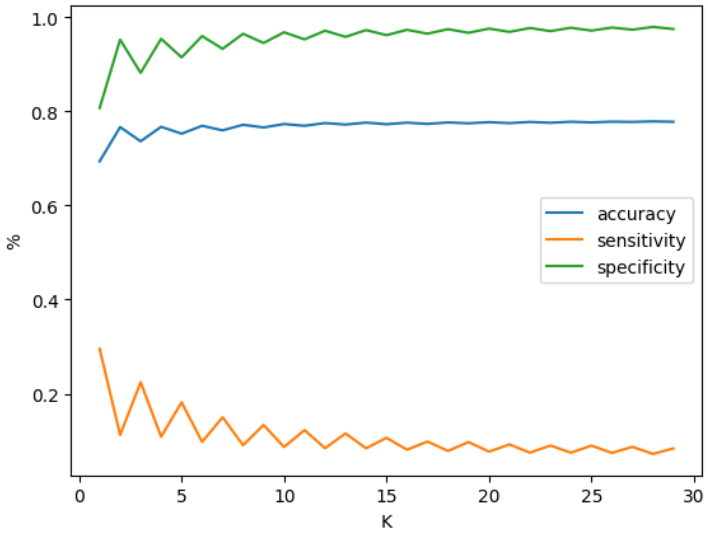
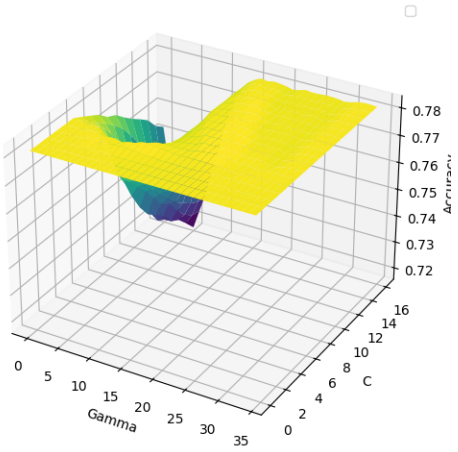


Figura 13: Estimação do melhor K para o classificador K-NN

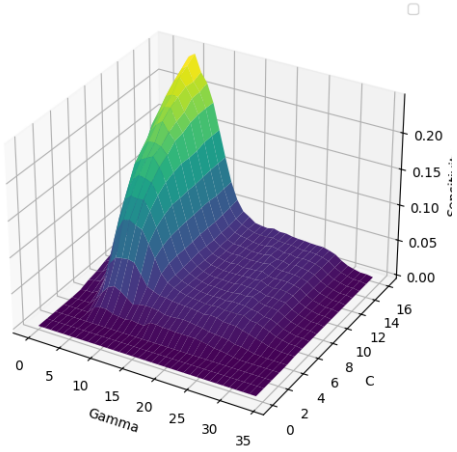
Devido à natureza do *dataset* observa-se que continuamos com dificuldade em manter uma boa sensibilidade. Sendo assim, fixaremos o $K = 1$ de forma a tentar maximizar a sensibilidade.

De forma a obter os resultados presentes na figura 14, lancei uma *SVM* para todas as combinações de $C \in [2^{-5}, 2^{12}[$ e $\gamma \in [2^{-30}, 2^{5}[$.

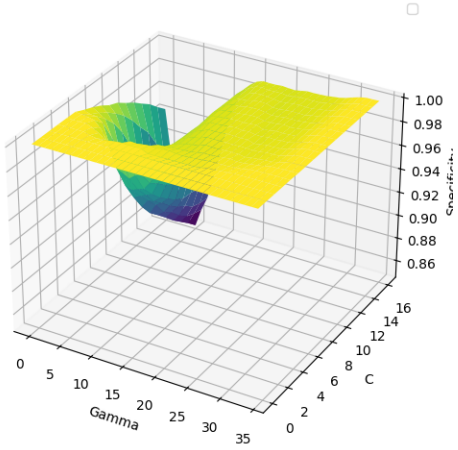
Utilizei 50% do *dataset* para treino e o restante para teste. Embora esteja ciente que estes resultados estão sujeitos a erro, o objetivo é estimar os melhores parâmetros da melhor forma possível, tendo em conta a complexidade do treino da *SVM*. Ainda assim, este teste durou cerca de seis horas.



(a) Estimação dos melhores parâmetros para a SVM - exatidão



(b) Estimação dos melhores parâmetros para a SVM - Sensibilidade

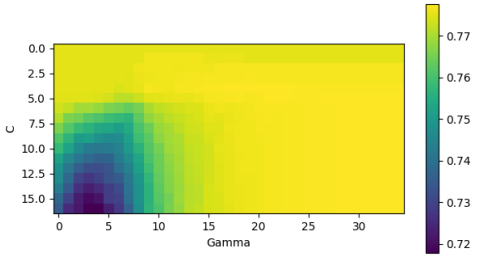


(c) Estimação dos melhores parâmetros para a SVM - Especificidade

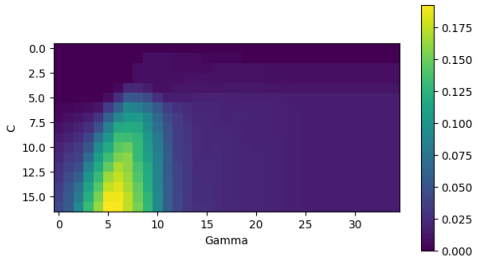
Figura 14: Estimação dos melhores parâmetros para a SVM com features da PCA

Como podemos ver pelas figuras 14a e 14b, a sensibilidade que o classificador ganha não é proporcional à perda da exatidão, ou seja, ao passo que ganhamos um pouco de sensibilidade iremos perder drasticamente na exatidão.

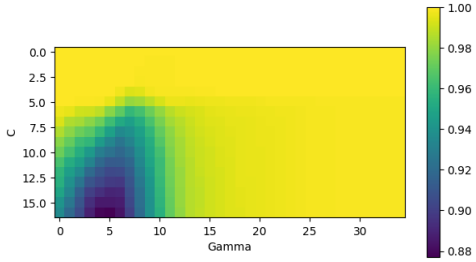
Para tornar essa análise mais compreensível, na Figura 15, projetei a superfície da Figura anterior no plano 2D.



(a) Estimação dos melhores parâmetros para a SVM - exatidão



(b) Estimação dos melhores parâmetros para a SVM - Sensibilidade



(c) Estimação dos melhores parâmetros para a SVM - Especificidade

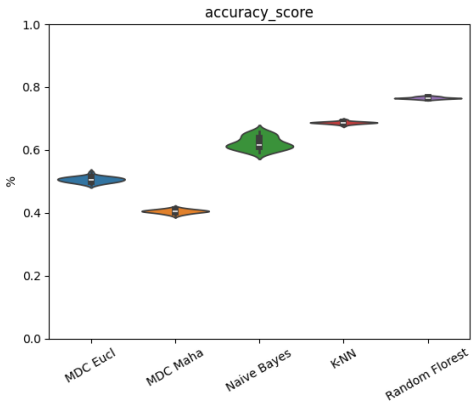
Figura 15: Estimação dos melhores parâmetros para a SVM com features da PCA

Após esta análise, tendo em consideração do potencial da sensibilidade e o custo computacional, não considero que seja vantajoso a *SVM* a este conjunto de *features*.

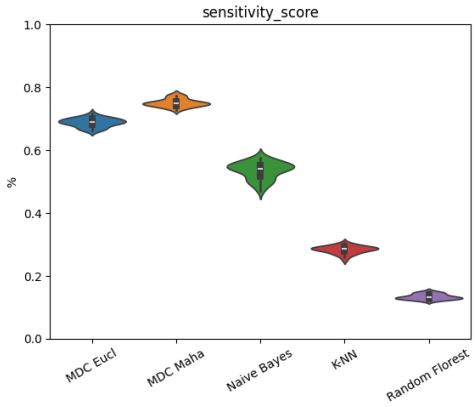
Na figura 16 apresento os resultados obtidos para os classificadores. Com estas *features* o classificador que obteve maior sensibilidade foi a *MDC* com a distância de *mahalanobis*, embora haja uma perda significativa de especificidade e de exatidão. No entanto, se quisermos uma boa performance em classificar os clientes que não serão capazes de pagar o empréstimo, este será o classificador ideal. Se admitirmos uma pequena perda de sensibilidade, com a *MDC euclidiana* conseguimos aumentar um pouco a exatidão e especificidade.

Considero que o classificador que manteve o melhor *trade off* entre as três métricas foi o *Nearest Bayes*, pois manteve uma exatidão e especificidade de 60% e uma sensibilidade de 50%.

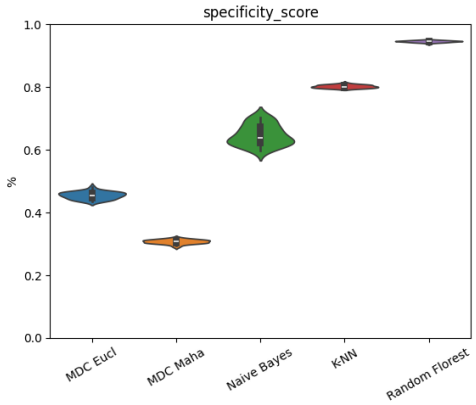
Os classificadores restantes tiveram uma perda de exatidão e sensibilidade sem qualquer diferença na sensibilidade, face aos resultados com os dados cruz.



(a) Exatidão dos classificadores com as features da PCA



(b) Sensibilidade dos classificadores com as features da PCA



(c) Especificidade dos classificadores com as features da PCA

Figura 16: Métricas de performance dos classificadores com as features da PCA

5.2.4. Classificação com AUC

Na figura 17 podemos ver os resultados obtidos para o valor de k, cujos foram obtidos da mesma forma descritos na secção anterior.

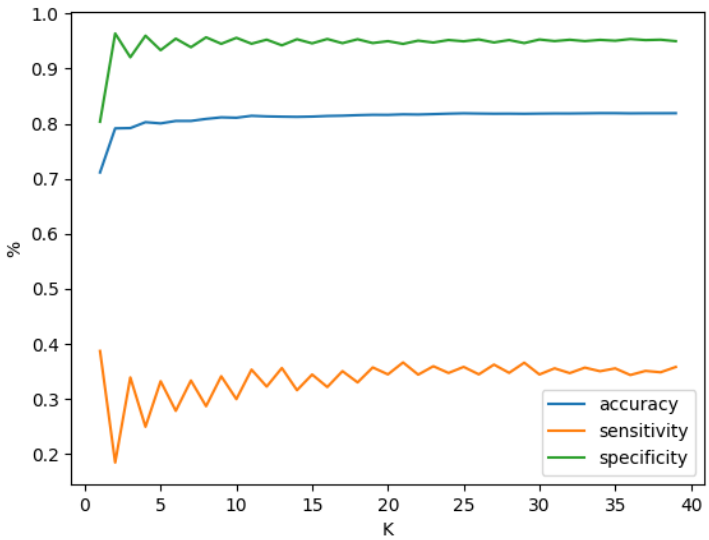
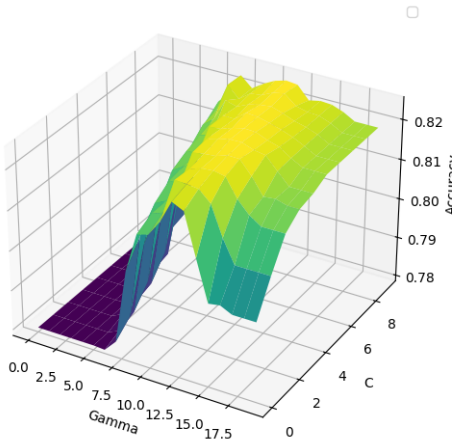


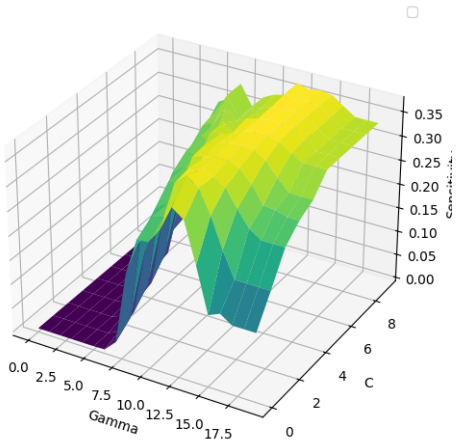
Figura 17: Estimação do melhor K para o classificador K-NN

Aqui podemos ver que para um $k = 1$ maximiza a sensibilidade. No entanto existe uma perda significativa da exatidão e da especificidade. Sendo assim irei fixar o $k = 20$, porque considero que seja o melhor equilíbrio entre as três métricas sem prejudicar a sensibilidade.

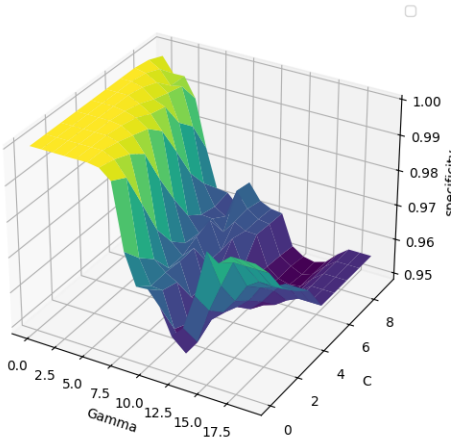
Para estimar os melhores valores para C e γ para as *SVM* implementei um *grid search* para os valores de C e γ, onde $C \in [2^{-5}, 2^{5}[$ e $\gamma \in [2^{-15}, 2^{5}[$ De forma a visualizar melhor os resultados criei a figura 18.



(a) Estimação dos melhores parâmetros para a SVM - exatidão



(b) Estimação dos melhores parâmetros para a SVM - Sensibilidade



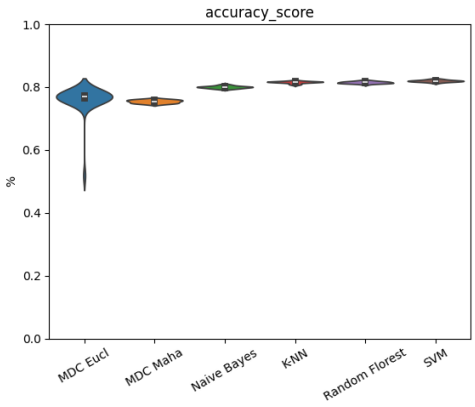
(c) Estimação dos melhores parâmetros para a SVM - Especificidade

Figura 18: Estimação dos melhores parâmetros para a SVM com features da AUC

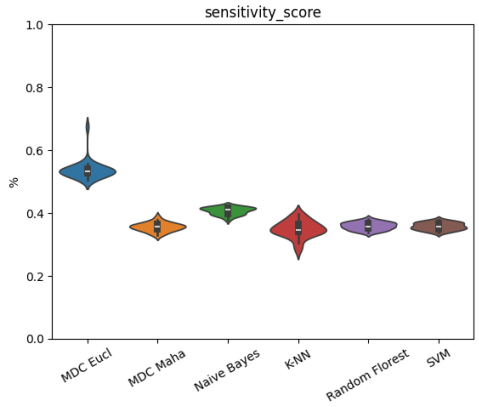
Novamente, enfrentamos dificuldades em obter uma boa sensibilidade. Os valores que maximizam a sensibilidade são $C = 2 \wedge \gamma = 0.25$.

Se analisarmos a figura 19a conseguimos observar que as *features* selecionadas pela *AUC* apresentam todos uma alta exatidão dentro do mesmo intervalo com baixo desvio padrão. O mesmo se pode dizer para a especificidade à exceção da *MDC* que apresenta um desvio padrão maior.

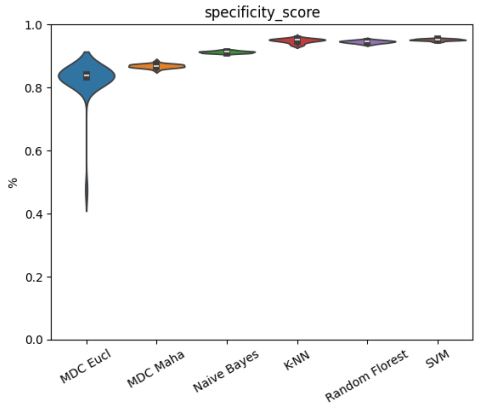
Em relação à sensibilidade, a *MDC* com distância euclidiana se destaca ligeiramente, enquanto os outros classificadores mantêm sensibilidade semelhante dentro do mesmo intervalo.



(a) Exatidão dos classificadores com as features da AUC



(b) Sensibilidade dos classificadores com as features da AUC



(c) Especificidade dos classificadores com as features da AUC

Figura 19: Métricas de performance dos classificadores com as features da AUC

Concluímos que as *features da AUC* permitem uma classificação mais equilibrada entre os classificadores. No entanto, a *MDC* euclidiana não se destaca o suficiente em comparação com as classificações das secções anteriores.

5.2.5. Classificação com KW

Na figura 20 podemos ver os resultados obtidos para o valor de k, cujos foram obtidos da mesma forma descritos na secção anterior.

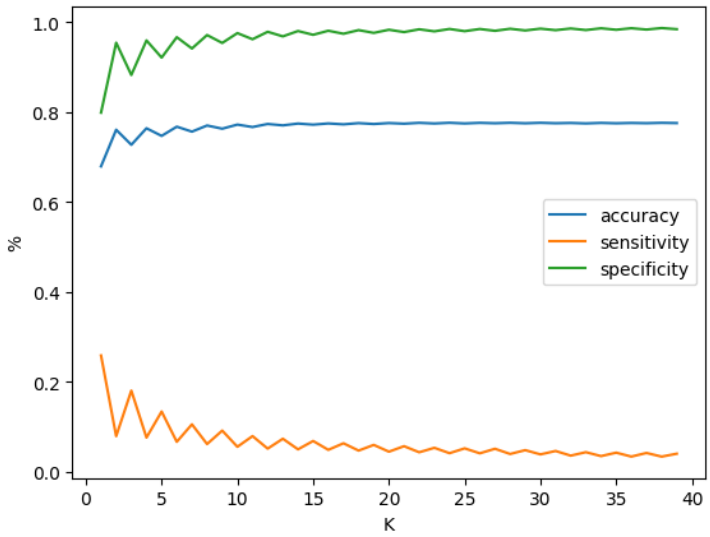
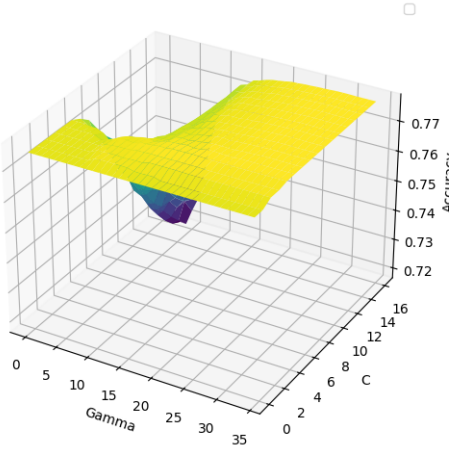


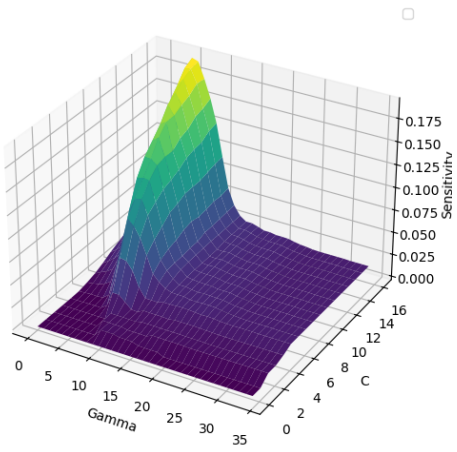
Figura 20: Estimação do melhor K para o classificador K-NN

Os resultados foram muito semelhantes aos da classificação com as *features da PCA*, e optei por fixar $K = 1$.

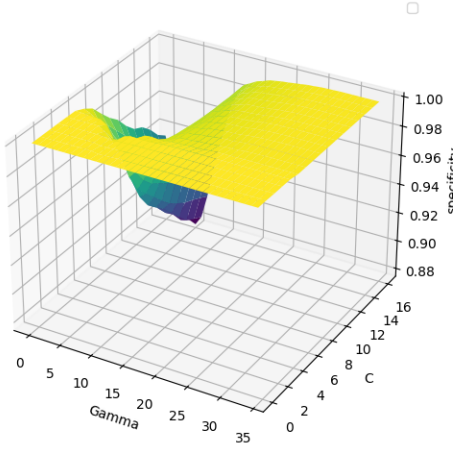
A abordagem do *grid search* para a estimação dos melhores parâmetros da *SVM* mantêm-se igual à estratégia usada para a classificação com as *features* da *PCA*. Podemos ver os resultados obtidos na figura 21.



(a) Estimação dos melhores parâmetros para a SVM - exatidão



(b) Estimação dos melhores parâmetros para a SVM - Sensibilidade



(c) Estimação dos melhores parâmetros para a SVM - Especificidade

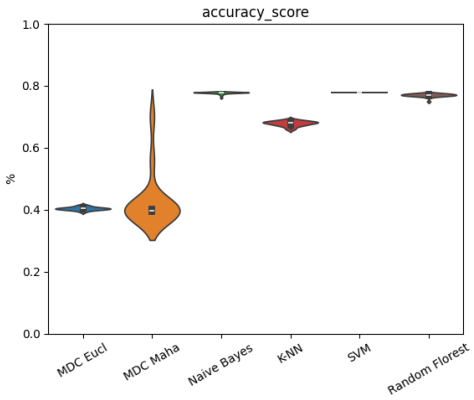
Figura 21: Estimação dos melhores parâmetros para a SVM com features do KW

Os resultados foram muito semelhantes aos da classificação com as *features da PCA*, mas ligeiramente melhores. Por esse motivo, decidi realizar a classificação com a *SVM*. Os melhores parâmetros calculados foram $C = 2048$ e $\gamma = 2.98e^{-08}$.

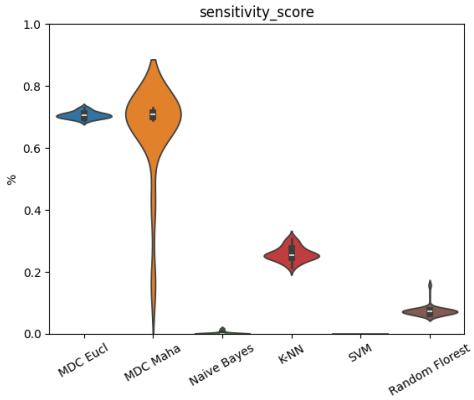
Vejamos os resultados das classificações na figura 22.

Existe claramente um destaque na sensibilidade por parte dos classificadores *MDC*, com uma performance muito parecida à classificação com a projeção da *PCA*.

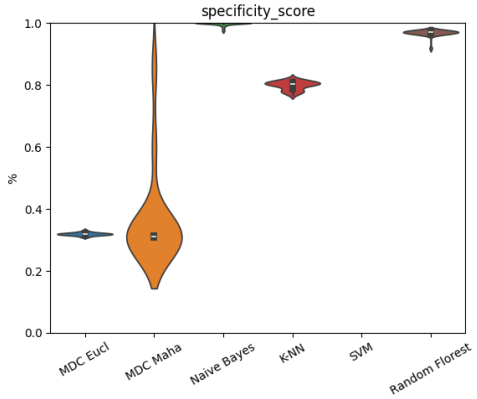
No entanto, o *Naive Bayes* e a *SVM* apresentam uma sensibilidade de 0%. Isso já era esperado, visto que visto que o classificador tem em conta a distribuição e como vimos na *EDA* estas *Features* estão sobrepostas. Já a *SVM* parece que fomos induzidos a erro. Era de esperar que uma coisa destas acontecesse visto que estávamos a admitir erros no estimação de parâmetros.



(a) Exatidão dos classificadores com as features da KW



(b) Sensibilidade dos classificadores com as features da KW

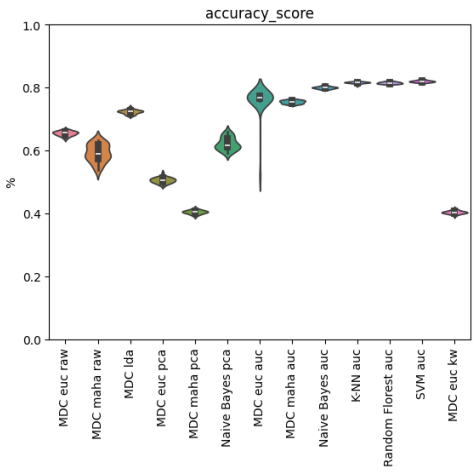


(c) Especificidade dos classificadores com as features da KW

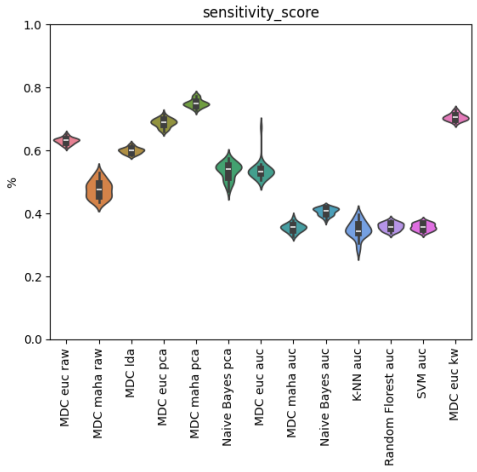
Figura 22: Métricas de performance dos classificadores com as features da KW

5.2.6. Classificadores mais relevantes

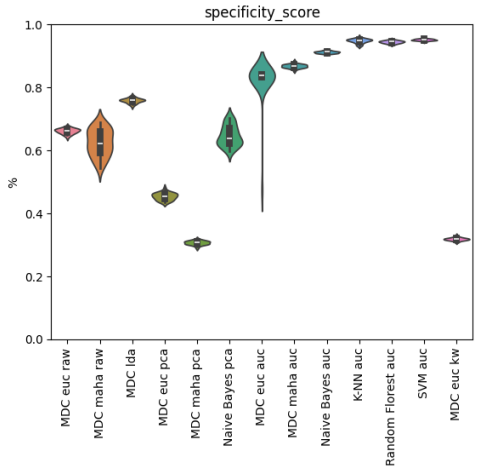
A figura 23 foi criada de forma a que o leitor possa comparar as métricas dos classificadores mais relevantes mais facilmente.



(a) Exatidão dos classificadores



(b) Sensibilidade dos classificadores



(c) Especificidade dos classificadores

Figura 23: Métricas de performance dos classificadores mais relevantes

Podemos ver que temos classificadores mais equilibrados:

- MDC com distância euclidiana e dados cruz
- MDC com distância de Mahalanobis e dados cruz
- Fisher LDA
- Naive Bayes com PCA

temos alguns que apresentam alta especificidade:

- MDC com distância euclidiana e com features da AUC
- K-NN com features da AUC
- Naive Bayes com features da AUC
- Random Florest com features da AUC
- SVM com features da AUC

e outros com alta sensibilidade:

- MDC com distância euclidiana e com features do KW

- MDC com distância de Mahalanobis e com features da PCA

6. Conclusão

Após uma análise abrangente dos resultados de diferentes classificadores com diversas técnicas de redução e seleção de *features*, podemos tirar algumas conclusões significativas.

Primeiramente, observamos que a escolha da técnica de redução e de seleção de *features* adequada, desempenham um papel crucial na performance dos classificadores. Tanto a *PCA* quanto a *AUC* e o *KW* oferecem diferentes perspetivas sobre os dados, cujas impactam significativamente o desempenho dos classificadores.

Alguns classificadores destacam-se em termos de equilíbrio entre exatidão, sensibilidade e especificidade, como a *MDC* com dados cruz, bem como o *Naive Bayes* com *PCA* e *Fisher LDA*. Esses modelos demonstram uma capacidade razoável de discriminar entre as classes positivas e negativas.

Por outro lado, outros classificadores destacam-se em termos de sensibilidade ou especificidade, dependendo da técnica de redução de *features* utilizada. Por exemplo, os classificadores treinados com *features da AUC* mostram uma alta especificidade.

É importante notar que a escolha do melhor classificador depende dos objetivos específicos do problema em questão. Se o foco estiver na identificação precisa de casos em que o cliente não é capaz de pagar, os classificadores com alta sensibilidade podem ser preferidos.

Em resumo, a seleção do classificador ideal depende de uma análise cuidadosa das necessidades e requisitos do problema em questão, bem como da compreensão das características e limitações de cada técnica de redução e seleção de *features*. A escolha adequada pode resultar num modelo eficaz e preciso para a classificação dos dados.