Atividade 1 Aprendizado Supervisionado: Relatório de Atividade

Julio Vinicius Amaral Oliveira
Universidade Estadual de Campinas (UNICAMP)
j230537@dac.unicamp.br

***Resumo*—O objetivo do trabalho é aplicar técnicas básicas de aprendizado supervisionado para detectar fraudes em transações financeiras. O trabalho foi dividido em duas partes: a primeira parte vamos implementar dois modelos de detecção de fraude, enquanto a segunda parte resolveremos algumas questões teóricas relacionadas ao tema da disciplina.**

I. Introdução

O dataset utilizado é o Credit Card Fraud Detection. Ele possui transações realizadas com cartões de crédito em setembro de 2013. O conjunto de dados possui a maioria das features anonimizadas, com exceção de algumas variáveis como: tempo, valor da transação e a variável que indica se a transação é fraude ou não. Uma característica bem importante desse datasete é que ele é muito desbalanceado, com apenas 0.172% das transações sendo fraudes.

II. Metodologia (Parte 1)

A. Análise Exploratório de Dados (EDA) e Pré-processamento

Como dito anteriormente, o dataset é bastante desbalanceado, o primeiro passo realizado foi verificar a quantidade de fraudes e não fraudes utilizando o método `value_counts()`. Obtendo o seguinte resultado:

Tabela I
Correlação das features com Class

Feature	Correlação com Class
V17	-0.326481
V14	-0.302544
V12	-0.260593
V10	-0.216883
V16	-0.196539
V3	-0.192961
V7	-0.187257

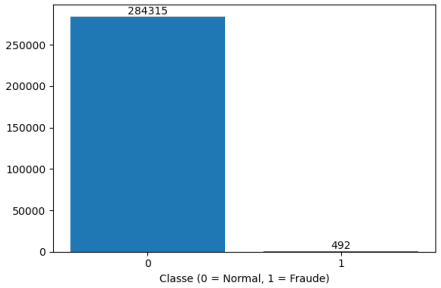


Figura 1. Proporção de fraudes e não fraudes no dataset

Após isso, procuramos entender como as features se relacionavam com a coluna de Class, que era nossa coluna alvo. Então usamos o método `corr()` para buscar a correlação entre Class e o restante das features. obtendo o seguinte resultado para as features com maior correlação absoluta:

B. Abordagem 1: Modelagem da Classe Normal (Detecção de Anomalia)

Para a primeira abordagem utilizamos o modelo de detecção de anomalias. Primeiramente,

separamos apenas as 3 features mais relevantes para o modelo, esse número foi escolhido com base na performance do modelo. Foi percebido que ao diminuir cada vez mais o número de features tanto a medida de AUC-PR quanto a matriz de confusão resultavam em valores melhores. Após isso, separamos 2000 transações normais, e com o StandartScaler aplicamos uma normalização para esses dados. Depois aplicamos a normalização para o conjunto inteiro de dados (Com exceção da coluna Class). Após isso, tentamos encontrar o melhor valor de bandwith usando o método `GridSearchCV`. O melhor valor encontrado foi 0.479, com isso aplicamos um KDE e calculamos o score de anomalias.

Tabela II
Transações normais

	anomaly_score	true_class
count	284315.000000	284315.0
mean	3.818309	0.0
std	4.005647	0.0
min	2.346846	0.0
25%	2.822756	0.0
50%	3.311860	0.0
75%	4.210312	0.0
max	557.518645	0.0

Tabela III
Transações fraudulentas

	anomaly_score	true_class
count	492.000000	492.0
mean	162.766740	1.0
std	250.679879	0.0
min	2.780611	1.0
25%	26.711496	1.0
50%	68.714390	1.0
75%	156.108730	1.0
max	1377.747682	1.0

Apos isso, tentamos encontrar o melhor limiar para separar as transações utilizando o método `precision_recall_curve`. Obtivemos que o melhor limiar para maximizar f1 era 34.9. Com isso, conseguimos os seguintes resultados após plotar a matriz de confusão:

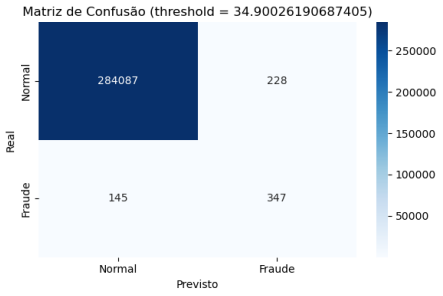


Figura 2. Matriz de confusão

Segundo o próprio repositório do Kaggle, a matriz de confusão não é o melhor indicador de desempenho quando temos classes desbalanceadas e nesses casos é recomendado o uso da área abaixo da curva de precision-recall. Cujo resultado foi de 0.623

C. Abordagem 2: Classificação Supervisionada

Nessa segunda abordagem utilizamos um modelo de regressão logística, utilizamos o método `train_test_split` do sklearn para separar 80% dos dados para treino e 20% para teste, isso foi realizado nos dados já normalizado com a ajuda do StandartScaler. Após isso, aplicamos o modelo de regressão logística e aplicamos o modelo nos dados de testes, obtendo a seguinte AUC-PR: 0.742, que foi um valor muito maior do que o obtido na primeira abordagem. Para ilustrar também o resultado, plotamos a matriz de confusão:

Tabela IV
Métricas de avaliação do modelo para as classes 0 (normal) e 1 (fraude)

	Classe 0	Classe 1
Precision	0.999	0.829
Recall	1.000	0.643
F1-score	1.000	0.724
Support	56864	98

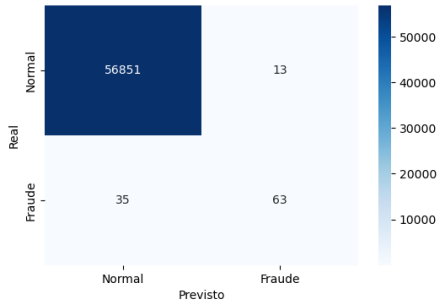


Figura 3. Matriz de confusão sem threshold

Para melhorar ainda mais o resultado, resolvemos encontrar um threshold utilizando o método `precision_recall_curve`, e ao fazer isso encontramos um valor melhor de recall, entretanto o valor da precision diminuiu um pouco. Mas dado o contexto que o modelo deveria ser usado, ter um recall alto é melhor pois evita perdas financeiras para a empresa que utiliza o modelo.

Tabela V
Métricas de avaliação do modelo para as classes 0 (normal) e 1 (fraude)

	Classe 0	**Classe 1**
Precision	0.999	0.692
Recall	0.999	0.827
F1-score	1.000	0.753
Support	56864	98

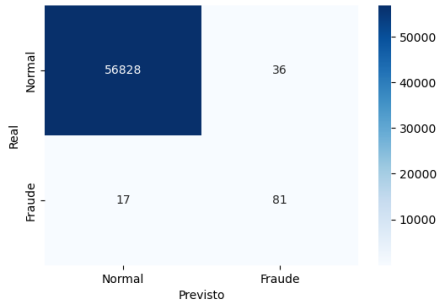


Figura 4. Matriz de confusão com threshold

D. Uso de LGBM

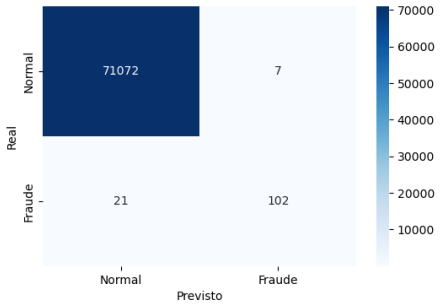
Ao utilizar o modelo LGBM, obtivemos um resultado muito superior aos anteriores. Calculamos os scores utilizando o `predict_proba` e ajustamos o threshold para maximizar o F1-score. Os thresholds, precision e recall foram encontrados utilizando o método `precision_recall_curve` e calculamos o F1 através da fórmula:

$$F1 = \frac{2 \cdot (precision \cdot recall)}{precision + recall}$$

Seguem os resultados de precisão, recall e F1-score:

Tabela VI
Métricas de avaliação do LGBM para as classes 0 (normal) e 1 (fraude)

	Classe 0	**Classe 1**
Precision	1.000	0.936
Recall	1.000	0.829
F1-score	1.000	0.879
Support	71079	123



Também calculamos a AUC-PR, que foi de 0.8585 A utilização do LGBM trouxe uma distribuição muito heterogênea entre os scores, oque pode ser visto na figura 5. Enquanto os scores de dados normais se concentram em torno de 0, os scores de dados fraudulentos se concentram em torno de 1, mostrando que o LGBM conseguiu separar bem as classes.

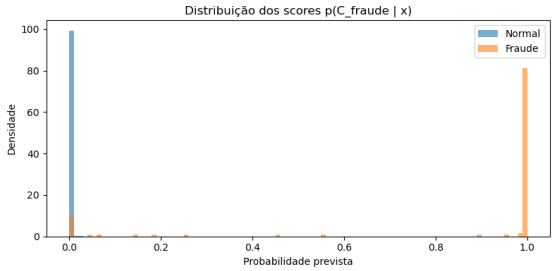


Figura 5. Distribuição dos scores do modelo LGBM

III. Discussão e Conclusão

A primeira abordagem, que utilizou o modelo de detecção de anomalias, obteve um resultado de AUC-PR de 0.623, enquanto a segunda abordagem, que utilizou o modelo de regressão logística, obteve um resultado de AUC-PR de 0.742. Por fim, a terceira abordagem, que utilizou o modelo LGBM, obteve um resultado de AUC-PR de 0.8585. Isso mostra que o modelo LGBM foi o mais eficaz para detectar fraudes no dataset. Deve-se levar em conta também, que na primeira abordagem, o resultado só foi tão alto, devido ao fato de que reduzimos consideravelmente a quantidade de features, pois em testes prévios, ao utilizar todas as features, os resultados eram muito ruins, com AUC-PR próximos de 0.1. A abordagem 2 pareceu melhor, tanto em questão de resultado quanto em questão de performance, já que para realizar a abordagem 1, tivemos que separar um conjunto inferior de dados, utilizar menos features para chegar em um resultado minimamente aceitável. Já na abordagem 2, utilizamos todas as features e mesmo assim chegamos em um resultado satisfatório. Quanto a escolha do limiar, utilizamos o método `precision_recall_curve` para encontrar o F1 com maior valor, em um cenário real, talvez o ideal fosse utilizar um limiar que favorecesse ainda mais o recall, dado que estamos lidando com fraudes que podem significar perda financeira para uma empresa, mas ao mesmo tempo é importante considerar que um modelo que tem uma precisão baixa pode causar problemas tanto para clientes que possam ter transações bloqueadas, causando insatisfação quanto para um time que precisará investigar essas transações.

IV. Respostas Teórico-Conceituais (Parte 2)

A. Exercício 1

1) Fronteira de Decisão com Covariâncias Diferentes e Priors Iguais: Sabemos que a probabilidade posterior de uma classe C_k dado um vetor de features $\mathbf{x}$ é dada pelo Teorema de Bayes:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}.$$

A fronteira de decisão é quando $p(C_1|\mathbf{x}) = p(C_2|\mathbf{x})$. Segundo o enunciado, $p(C_1) = p(C_2) = 0.5$.

Sabemos que:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = p(\mathbf{x}|C_k)$$
$$= \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

Igualando $p(\mathbf{x}|C_1) = p(\mathbf{x}|C_2)$

$$\frac{p(C_1|\mathbf{x})p(\mathbf{x})}{p(C_1)} = \frac{p(C_2|\mathbf{x})p(\mathbf{x})}{p(C_2)}$$

Ao simplificar, temos:

$$p(\mathbf{x}|C_1) = p(\mathbf{x}|C_2)$$

Portanto, igualando a expressão obtida acima e aplicando o logaritmo natural:

$$-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_1|$$
$$= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_2|$$

Dividindo por -1/2 e rearranjando, a equação fica:

$$(\mathbf{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \ln|\boldsymbol{\Sigma}_1|$$
$$= (\mathbf{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \ln|\boldsymbol{\Sigma}_2|.$$

Como $x^T\Sigma^{-1}\mu$ é escalar, $x^T\Sigma^{-1}\mu = (x^T\Sigma^{-1}\mu)^T = \mu^T\Sigma^{-1}x$, expandindo os termos e aplicando essa propriedade, temos:

$$\mathbf{x}^T\boldsymbol{\Sigma}_1^{-1}\mathbf{x} - 2\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\mathbf{x} + \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \ln|\boldsymbol{\Sigma}_1|$$
$$= \mathbf{x}^T\boldsymbol{\Sigma}_2^{-1}\mathbf{x} - 2\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\mathbf{x} + \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 + \ln|\boldsymbol{\Sigma}_2|.$$

Reorganizando os termos, temos:

$$\mathbf{x}^T(\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1})\mathbf{x} - 2(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1})\mathbf{x} + \left(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 + \ln|\boldsymbol{\Sigma}_1| - \ln|\boldsymbol{\Sigma}_2|\right) = 0.$$

Essa equação tem um formato de $x^TAx + b^Tx + c = 0$ que é uma superfície quadrática.

2) Efeito da Alteração do Prior com Covariâncias Compartilhadas: Como os priors são $p(C_1) = \pi$ e $p(C_2) = 1 - \pi$, tomando a condição para a fronteira de decisão $p(\mathbf{x}|C_1)p(C_1) = p(\mathbf{x}|C_2)p(C_2)$ temos que:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})\pi = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})(1 - \pi).$$

Aplicando o logaritmo natural nos dois lados:

$$-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln\pi = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_2) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln(1 - \pi).$$

Expandindo os termos, temos:

$$-\frac{1}{2}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1) + \ln\pi = -\frac{1}{2}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2) + \ln(1 - \pi)$$

Simplificando e rearranjando, temos:

$$(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2) + \ln\left(\frac{\pi}{1 - \pi}\right) = 0.$$

Essa expressão tem o formato de uma equação:

$$\omega^T x + \omega_0 = 0$$

$$\omega_0 = -\frac{1}{2}(\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2) + \ln\left(\frac{\pi}{1 - \pi}\right)$$

$$\omega = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

$\ln\left(\frac{\pi}{1-\pi}\right)$ é um deslocamento escalar na equação linear. Alterar o prior π resulta no deslocamento paralelo da fronteira de decisão.

- Se $\pi > 0.5$ ($p(C_1) > p(C_2)$), implica que $\ln\left(\frac{\pi}{1-\pi}\right) > 0$. Para satisfazer a equação, $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}\mathbf{x}$ precisa ser menor, fazendo com que a fronteira se expanda para a região de C_1.
- Se $\pi < 0.5$ (ou seja, $p(C_1) < p(C_2)$), então $\ln\left(\frac{\pi}{1-\pi}\right) < 0$, e a fronteira se desloca na direção oposta, expandindo a região de C_2.
- Se $\pi = 0.5$, o termo logarítmico é zero, e a fronteira não se desloca para nenhum dos lados, sendo definida pelas médias e covariância.

3) Uso da Distância de Mahalanobis para Classificação: A distância de Mahalanobis pode ser usada para classificação comparando a distância de um ponto $\mathbf{x}$ para os centros das classes ($\boldsymbol{\mu}_1$ e $\boldsymbol{\mu}_2$), ponderadas pelas suas respectivas matrizes de covariância. A regra de classificação seria classificar $\mathbf{x}$ como pertencente à classe C_1 para a qual a distância de Mahalanobis quadrática Δ_1^2 é mínima. Ou seja:

- Se $\Delta_1^2 \leqslant \Delta_2^2$, classificar $\mathbf{x}$ como C_1 (Normal).
- Se $\Delta_1^2 > \Delta_2^2$, classificar $\mathbf{x}$ como C_2 (Fraude).

Premissas Implícitas:

- As classes devem seguir uma distribuição Gaussiana multivariada.
- Os parâmetros $\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2$ são conhecidos ou foram bem estimados.
- Priors iguais para as classes $p(C_1) = p(C_2)$.

B. Exercício 2: Verossimilhança, entropia cruzada e regressão logística

1) Mostrar que maximizar a função de verossimilhança é equivalente a minimizar a entropia cruzada: Para um modelo de regressão logística com alvos binários:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}$$

Sabendo que maximizar a função é o mesmo que maximizar o logaritmo natural, podemos aplicar o logaritmo natural para que a função se pareça mais com o nosso objetivo:

$$\ln p(\mathbf{t}|\mathbf{w}) = \sum_{n=1}^{N} \ln\left(y_n^{t_n}(1 - y_n)^{1-t_n}\right).$$

Usando as propriedades do logaritmo:

$$\ln p(\mathbf{t}|\mathbf{w}) = \sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right].$$

A função de erro de entropia cruzada é definida por:

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right]$$

Comparando as duas equações, vemos que $E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w})$. Portanto, maximizar a verossimilhança é equivalente a minimizar a entropia cruzada.

2) Derivando $\nabla E(\mathbf{w})$: Lembrando que $y_n = \sigma(\mathbf{a}_n)$, com $\mathbf{a}_n = \mathbf{w}^T \boldsymbol{\phi}_n$, e que

$$\frac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a)),$$

a função de erro é:

$$\begin{aligned} E(\mathbf{w}) = -\sum_{n=1}^{N} [& t_n \ln \sigma(\mathbf{w}^T \boldsymbol{\phi}_n) \\ & + (1 - t_n) \ln(1 - \sigma(\mathbf{w}^T \boldsymbol{\phi}_n))]. \end{aligned}$$

O gradiente de $E(\mathbf{w})$ é:

$$\begin{aligned} \nabla E(\mathbf{w}) &= -\sum_{n=1}^{N} [t_n \frac{1}{y_n} y_n(1 - y_n)\boldsymbol{\phi}_n \\ &\quad + (1 - t_n) \frac{1}{1 - y_n} (-y_n(1 - y_n))\boldsymbol{\phi}_n] \\ &= -\sum_{n=1}^{N} \left[t_n(1 - y_n) - (1 - t_n)y_n \right] \boldsymbol{\phi}_n \\ &= \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n. \end{aligned}$$

3) Explicação do problema de overfitting: O overfitting ocorre quando os dados são linearmente separáveis. Nesse caso, existe um $\mathbf{w}$ tal que $\mathbf{w}^T \boldsymbol{\phi}_n$ separa perfeitamente as classes, fazendo com que $\|\mathbf{w}\| \to \infty$.

Para evitar isso, usa-se regularização L_2:

$$\begin{aligned} E_{\text{regularizado}}(\mathbf{w}) = -\sum_{n=1}^{N} & [t_n \ln y_n \\ & + (1 - t_n) \ln(1 - y_n)] + \frac{\lambda}{2} \|\mathbf{w}\|^2, \end{aligned}$$

O termo $\frac{\lambda}{2} \|\mathbf{w}\|^2$ penaliza pesos grandes, fazendo com que o modelo faça um overfitting.

C. Exercício 3: Decomposição viés-variância

1) Explicação $(bias)^2$, variance e noise:

$Bias^2$: O viés quadrático mede o erro sistemático do modelo, ou seja, o quanto em média o modelo se difere do valor real. Um modelo com um viés alto não consegue entender a complexidade dos dados, levando a um underfitting.

Variance: A variância mede o quanto as nossas predições pra um dado ponto variam usando diferentes conjuntos de treino. Um modelo que tem alta variância é muito sensível às variações nos dados de treino, fazendo com que o modelo não consiga performar bem em dados que ele não conhece, levando a um overfitting.

Noise: O ruído é a parte do erro que não é possível de ser reduzida, pois ele é uma característica inerente aos dados.

2) Avaliação de comportamento dos modelos de complexidade variável: Para essa tarefa foi escolhido o dataset California Housing, para que o modelo fosse treinado mais rapidamente, separamos apenas 3 features e utilizamos regressão polinomial, com o grau variando de 0 a 6, KMN para 9 valores diferentes de k pertencentes ao intervalo de 1 a 60 e ridge com valores de $\alpha \in \{10^{-5}, 10^{-4}, 10^{-3}, \ldots, 10^{3}\}$. Ao final, plotamos um gráfico do erro quadrático médio em função da complexidade do modelo.

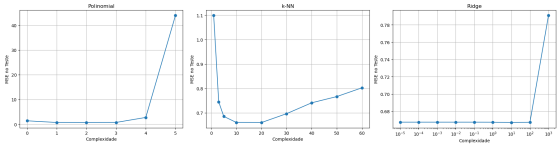


Figura 6. Erro quadrático médio em função da complexidade de cada modelo

Com isso, podemos ver que quando os modelos são mais simples, temos um um viés maior, resultando em um erro maior, ao caminhar sobre a complexidade vamos obtendo um valor intermediário de erro que vai aumentando conforme a complexidade do modelo vai aumentando, aumentando assim a variância.

Avaliação de viés e variância: Utilizamos 30 amostras bootstraps e um modelo de regressão que variava do grau 0 a 14, para cada grau, calculamos o viés e a variância, obtendo o gráfico abaixo:

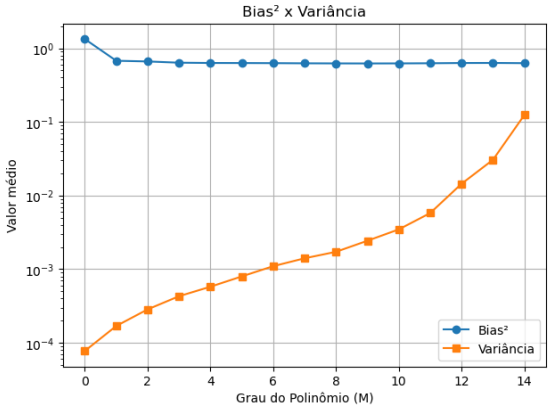


Figura 7. Viés e variância em função da complexidade

Podemos ver que o viés diminui quando a complexidade do modelo aumenta, enquanto a variância aumenta a passos largos, oque condiz com a teoria, já que quanto mais o modelo vai ficando complexo, mais ele vai se ajustando aos dados de treino, aumentando a variância.

D. Exercício 4: Avaliação em dados desbalanceados

1) Acurácia padrão em datasets desbalanceados: Em dados desbalanceados, a acurácia padrão pode dar uma falsa sensação que um modelo está muito bom. Já que um classificador simples que sempre classifica as transações como a classe majoritária pode ter uma alta acurácia pois ele irá classificar corretamente na grande maioria das vezes. Isso acontence porque a acurácia é definida como:

$$\text{Acurácia} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Em datasets desbalanceados, a classe majoritária tem um peso muito grande sobre essa métrica. Sendo assim, o classificador não cumpriria seu objetivo que é identificar fraudes, porém ainda sim teria uma acurácia alta.

2) Precisão, Recall e F1-Score:

Precision: Parcela das transações classificadas como fraude e que de fato são fraude:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Em que:

- TP: quantidade de fraudes classificadas corretamente.
- FP: quantidade de transações normais classificadas como fraude.

Recall: Parcela de todas as transações que de fato eram fraudes que foram classificadas corretamente:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Em que:

- FN: quantidade de fraudes que foram classificadas como transações normais.

F1-Score: Média harmônica entre precisão e recall:

$$F_1 = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Quando priorizar Recall sobre Precisão: Em situações em que perder uma fraude gera prejuízo financeiro para a empresa que utiliza o classificador ou em cenários em que uma classificação errada pode gerar muito prejuízo para o utilizador, como em resultado de doenças.

Quando priorizar Precisão sobre Recall: Em situações em que o valor de falsos positivos é alto, gerando algum esforço manual ou insatisfação do cliente para com os serviços oferecidos pela empresa que utiliza o classificador. Além disso, para que a precisão tenha mais relevância que o

recall, é necessário que o impacto de um falso negativo não seja tão alto para o negócio. Por exemplo, em um sistema de identificação de câncer, um falso negativo poderia levar a um diagnóstico mais tardio da doença.

3) Curva Precisão–Recall e AUC–PR: O procedimento para calcular a curva PR seria:

1) Ordenar todos os exemplos em ordem decrescente dos seus scores s_n.
2) Para cada limiar τ:
   - Classificar x_n como fraude se $s_n \geq \tau$, caso contrário, classificar como normal.
   - Calcular

$$\text{Precision}(\tau) = \frac{\#\{n : s_n \geq \tau,\ t_n = 1\}}{\#\{n : s_n \geq \tau\}},$$

$$\text{Recall}(\tau) = \frac{\#\{n : s_n \geq \tau,\ t_n = 1\}}{\#\{n : t_n = 1\}}.$$

3) Plotar cada ponto do par ($Recall(\tau)$, $Precision(\tau)$) e formar a curva.

A área Sob a Curva PR é dada por

$$\text{AUC-PR} = \int_0^1 \text{Precision}(r)\, \mathrm{d}r,$$

em que Precision(r) é a precisão correspondente ao recall r.

a) Por que AUC–PR é preferível à ROC em cenários muito desbalanceados: A curva ROC avalia a taxa de verdadeiros positivos e falsos positivos, em casos desbalanceados a taxa de falsos positivo

$$FPR = \frac{FP}{FP + TN}$$

Permanece baixa pois a quantidade de verdadeiro negativos é muito alta, então essa métrica mascara um modelo ruim. Enquanto a AUC-PR avalia diretamente os dados de fraude, ou seja, não temos um denominador muito alto que mascara o valor da métrica.

Referências

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.

[3] Kaggle, "Credit Card Fraud Detection," Kaggle, [Online]. Disponível em: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. [Acesso em: 14-maio-2025].

Apêndice

Tabela VII
Correlação das features com a variável Class

Feature	**Correlação com Class**
V17	-0.326481
V14	-0.302544
V12	-0.260593
V10	-0.216883
V16	-0.196539
V3	-0.192961
V7	-0.187257
V11	0.154876
V4	0.133447
V18	-0.111485
V1	-0.101347
V9	-0.097733
V5	-0.094974
V2	0.091289
V6	-0.043643
V21	0.040413
V19	0.034783
V20	0.020090
V8	0.019875
V27	0.017580
Time	-0.012323
V28	0.009536
V24	-0.007221
Amount	0.005632
V13	-0.004570
V26	0.004455
V15	-0.004223
V25	0.003308
V23	-0.002685
V22	0.000805