

Aprendizado Semi-Supervisionado Aplicado a Grandes Volumes de Dados Não Normalizados Provenientes de Fraudes em Cartões de Crédito

João Pedro Augusto Costa

Resumo—Operações com cartões de crédito tornaram-se rotineiras para pessoas de várias classes sociais ao longo de diversos países, se tornando a principal forma de pagamentos de bens e serviços [7]. A quantidade massiva de dados gerada por essas operações dificultam a identificação de fraudes, o que vem a causar prejuízos financeiros tanto para as operadoras de cartão quanto aos clientes. O objetivo desse trabalho é avaliar o uso de algoritmos de aprendizado semi-supervisionado em dados provenientes de transações reais com cartões de crédito, verificando sua taxa de acerto e comparando com abordagens de aprendizado supervisionado. O experimentos computacionais demonstraram que os algoritmos de aprendizagem semi-supervisionada obtêm melhor desempenho em todas as configurações testadas.

I. INTRODUÇÃO

Fraudes em cartões de crédito consistem em pagamentos não autorizados que podem partir de dados obtidos com clonagens, engenharia social, uso de cartões roubados ou perdidos e até mesmo mais de uma dessas possibilidades [1], [9].

De acordo com a pesquisa encontrada em [1], dentro de um grupo de 6035 pessoas entrevistadas em diversos países do mundo, cerca de 30% enfrentaram problemas de fraude nos últimos cinco anos. Dentro desse grupo, cerca de 17% relatou ter sido vítima de fraude mais de uma vez, isso apenas no ano de 2016.

O número elevado no uso de cartões de crédito, junto à prática de comportamentos considerados de risco por parte dos usuários [9] torna uma tarefa muito difícil identificar possíveis fraudes, dada a quantidade de dados gerada pelas transações.

No ano de 2016, México, Estados Unidos e Brasil destacaram-se de forma negativa em relação ao número de fraudes em cartões de crédito, com 56%, 49% e 47% de usuários que reportam fraudes em cartões de crédito respectivamente. A principal justificativa para números tão expressivos, se dá ao uso de cartões de forma rotineira tanto para compras online quanto presenciais e as condições inseguras a que se submetem esses usuários, como não utilização de cartões com chip, acesso a sites potencialmente inseguros e falta de utilização de softwares de antivírus em seus computadores pessoais.

II. MOTIVAÇÃO

Dados do mundo real podem não se encontrar de forma estruturada e pronta para uma análise de maneira mais rápida. Os dados geralmente estão em escalas diferentes, além de haverem variáveis não observadas para alguns registros.

Dados provindos de cartões operações com cartões de crédito possuem uma característica que dificulta ainda mais a análise, a confidencialidade. Esses dados devem ser utilizados mantendo a privacidade dos usuários, o que torna o uso de métodos para inserção e substituição [3] de variáveis essenciais na análise desse tipo de dado.

Mensurar a quantidade de dados provenientes de transações financeiras não é fácil, mas estima-se que esteja na faixa de bilhões por ano [1]. Dada a quantidade de dados em que a maioria dos registros é provinda de transações consideradas normais, tem-se o problema de desbalanço nos dados [2], o que faz com que o uso de algoritmos de aprendizagem supervisionada não possua um bom desempenho sem o uso de mecanismos para rebalancear os dados.

Dados os conceitos explicados nessa seção, os algoritmos de aprendizagem semi-supervisionada parecem promissores nesse tipo de problema. Os algoritmos de aprendizagem semi-supervisionada trabalham utilizando conceitos de agrupamento, fazendo com que a capacidade de generalização de instâncias que representam operações fraudulentas seja uma etapa essencial nesse processo.

III. METODOLOGIA

O principal objetivo deste trabalho é investigar os resultados obtidos pelos algoritmos de aprendizado semi-supervisionado quando utilizados para classificar instâncias em dados reais de transações com cartões de crédito.

Os dados utilizados foram cedidos por empresas de cartão de crédito europeias e são referentes a transações que ocorreram no mês de setembro de 2013 [2]. Os atributos do *dataset* estão em escalas diferentes e a distribuição de classes está desbalanceada. O *dataset* contém 284.807 transações, entre as quais estão incluídas 492 fraudes, apenas 0,172% do total. O *dataset* possui 28 atributos, sendo o último correspondente a classe, em que o problema é binário, sendo que 1 representa uma fraude e 0 caso contrário.

Para que a abordagem atenda aos objetivos, deve-se garantir que os atributos não estejam balanceados ou normalizados, utilizando técnicas para inserção de *missing data*, assim como de substituição dos dados que faltam.

Para garantir controle total na quantidade de *missing data* do *dataset* foram utilizados os métodos de inserção *MCAR*, *MUOV* e *MIV* encontrados em [3]. O *MCAR* é um algoritmo simples que escolhe de forma aleatória uma coordenada no *dataset* e substitui o atributo nessa posição. O procedimento é

Tabela II: Configurações

Missing Data	Substituição	Algoritmo	Nome Configuração
MCAR	MEAN	svm.SVC	mcar_mean_svm
MCAR	MEAN	MLPClassifier	mcar_mean_mlp
MCAR	MEAN	LabelSpreading	mcar_mean_ls
MCAR	MEAN	LabelPropagation	mcar_mean_lp
MUOV	MEAN	svm.SVC	muov_mean_svm
MUOV	MEAN	MLPClassifier	muov_mean_mlp
MUOV	MEAN	LabelSpreading	muov_mean_ls
MUOV	MEAN	LabelPropagation	muov_mean_lp
MIV	MEAN	svm.SVC	miv_mean_svm
MIV	MEAN	MLPClassifier	miv_mean_mlp
MIV	MEAN	LabelSpreading	miv_mean_ls
MIV	MEAN	LabelPropagation	miv_mean_lp
MCAR	MEDIAN	svm.SVC	mcar_median_svm
MCAR	MEDIAN	MLPClassifier	mcar_median_mlp
MCAR	MEDIAN	LabelSpreading	mcar_median_ls
MCAR	MEDIAN	LabelPropagation	mcar_median_lp
MUOV	MEDIAN	svm.SVC	muov_median_svm
MUOV	MEDIAN	MLPClassifier	muov_median_mlp
MUOV	MEDIAN	LabelSpreading	muov_median_ls
MUOV	MEDIAN	LabelPropagation	muov_median_lp
MIV	MEDIAN	svm.SVC	miv_median_svm
MIV	MEDIAN	MLPClassifier	miv_median_mlp
MIV	MEDIAN	LabelSpreading	miv_median_ls
MIV	MEDIAN	LabelPropagation	miv_median_lp
MCAR	MOST	svm.SVC	mcar_most_svm
MCAR	MOST	MLPClassifier	mcar_most_mlp
MCAR	MOST	LabelSpreading	mcar_most_ls
MCAR	MOST	LabelPropagation	mcar_most_lp
MUOV	MOST	svm.SVC	muov_most_svm
MUOV	MOST	MLPClassifier	muov_most_mlp
MUOV	MOST	LabelSpreading	muov_most_ls
MUOV	MOST	LabelPropagation	muov_most_lp
MIV	MOST	svm.SVC	miv_most_svm
MIV	MOST	MLPClassifier	miv_most_mlp
MIV	MOST	LabelSpreading	miv_most_ls
MIV	MOST	LabelPropagation	miv_most_lp

O *dataset* contendo as fraudes utilizado foi normalizado utilizando a Equação 3 e dividido na proporção 80% e 20% para treino e teste respectivamente. A divisão é feita de maneira aleatória, utilizando a função *train_test_split()* do Scikit-Learn, portanto o número de instâncias pertencentes a cada classe se dá de forma probabilística

$$X_n = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3)$$

A. Método de Substituição MEAN

No método de substituição *MEAN* os valores que faltam no *dataset* são substituídos pela média aritmética simples de todos os valores do atributo correspondente.

A Figura 2 exibe a distribuição para os dados de teste utilizando o método de inserção de *Missing Data MCAR* e o de substituição *MEAN*. O gráfico está organizado de maneira que o eixo das abscissas representa o número da instância no conjunto e o eixo das ordenadas é a sua classe. Nesse teste, 86 instâncias pertencem a classe 1 ou positivo para fraude, enquanto 56.961 são apresentadas transação negativas ou não

fraudulentas, seguindo a distribuição do *dataset* e apresentando desbalançamento.

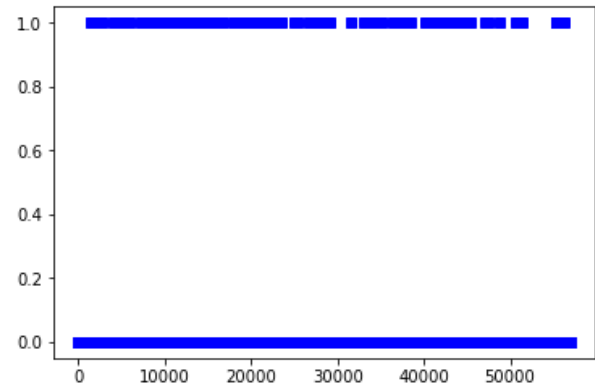


Figura 2: Distribuição Dados Testes mcar_mean_*

A Tabela III exibe os resultados obtidos por todos os algoritmos quando utilizados no *dataset* utilizando o método de inserção de *Missing Data MCAR* e o método de substituição *MEAN*. O Algoritmo de aprendizado semi-supervisionado *LabelPropagation* obteve o melhor resultado nesse cenário, seguido pelo também semi-supervisionado *LabelSpreading* e pela implementação do algoritmo de máquinas de vetores de suporte.

Tabela III: Resultados mcar_mean_*

Config	V. Posit	Acurácia	Distribuição
mcar_mean_svm	58	67,44%	
mcar_mean_mlp	57	66,27%	
mcar_mean_ls	58	67,44%	
mcar_mean_lp	62	72,09%	

A Figura 3 exibe a distribuição dos dados de teste utilizando o método de inserção *MUOV* e o de substituição *MEAN*. Nesse caso haviam 114 instâncias positivas para fraude.

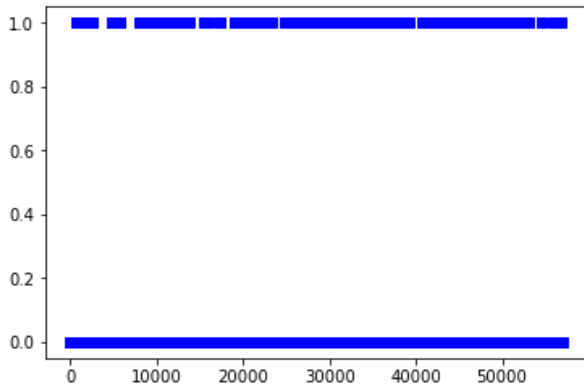


Figura 3: Distribuição Dados Testes muov_mean_*

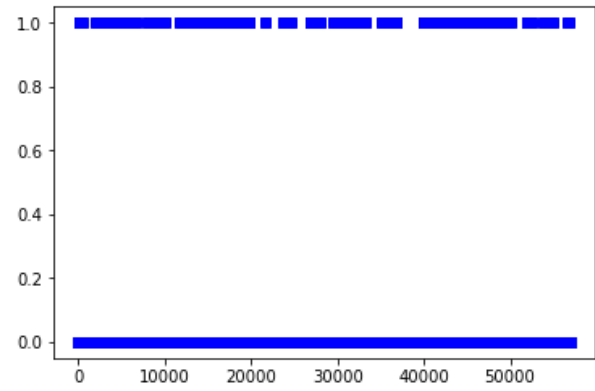


Figura 4: Distribuição Dados Testes miv_mean_*

Tabela IV: Resultados muov_mean_*

Config	V. Posit	Acurácia	Distribuição
muov_mean_svm	76	66,66%	
muov_mean_mlp	73	64,03%	
muov_mean_ls	94	82,45%	
muov_mean_lp	89	78,07%	

A Tabela IV mostra que os algoritmos de aprendizado semi-supervisionado novamente obtiveram melhores resultados que os algoritmos de aprendizado supervisionado. Com destaque para o algoritmo *LabelSpreading*, que obteve uma taxa de acerto de 82.45%.

Na Figura 4 é possível observar a distribuição para a configuração contendo *MIV* e *MEAN*, onde são encontradas 101 instâncias pertencentes a classe 1 ou positivo para fraude.

A Tabela V mostra que o algoritmo de aprendizagem *LabelSpreading* obteve os melhores resultados, seguido pelos algoritmos de aprendizagem supervisionada, enquanto o *LabelPropagation* ocupou a última posição.

Tabela V: Resultados miv_mean_*

Config	V. Posit	Acurácia	Distribuição
miv_mean_svm	64	63,36%	
miv_mean_mlp	64	63,36%	
miv_mean_ls	66	65,34%	
miv_mean_lp	49	48,51%	

Observando a Figura 5, nota-se que a acurácia obtida pelos algoritmos de aprendizado semi-supervisionado obtêm resultados melhores quando a quantidade de positivos nos dados de teste é maior, indicando que sua capacidade de generalização consegue uma boa generalização em dados altamente desbalanceados, desde que aja uma quantidade significativa de instâncias de ambas as classes.

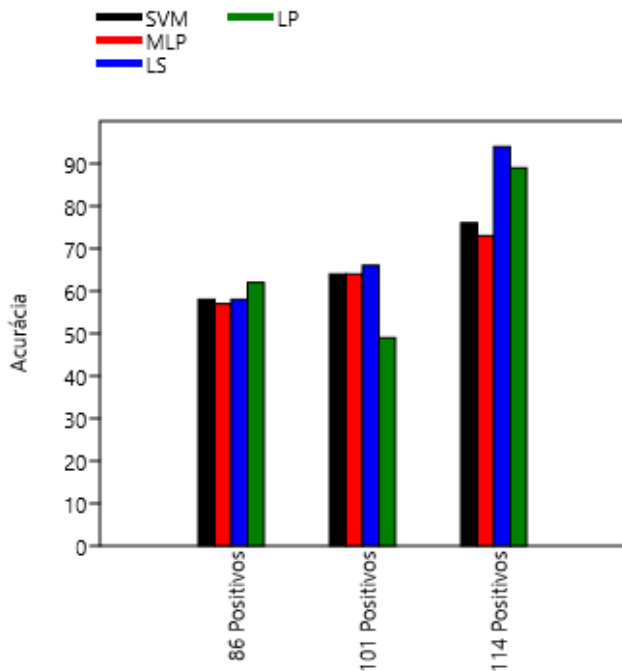


Figura 5: Desempenho *_mean_*

Tabela VI: Resultados mcar_median_*

Config	V. Posit	Acurácia	Distribuição
mcar_median_svm	67	68,36%	
mcar_median_mlp	62	63,26%	
mcar_median_ls	81	82,65%	
mcar_median_lp	72	73,46%	

B. Método de Substituição MEDIAN

No método de substituição *MEDIAN*, os valores de cada atributo são ordenados em ordem crescente, então os valores que faltam no *dataset* são substituídos pelo valor localizado no meio do arranjo ordenado do atributo correspondente.

A Figura 6 mostra as distribuição quando dos dados de teste utilizados para o método de inserção de *Missing Data MCAR* e de substituição *MEDIAN*. Nesse teste haviam 98 fraudes. A Tabela VI mostra que os algoritmos de aprendizado semi-supervisionado obtiveram os melhores resultados novamente.

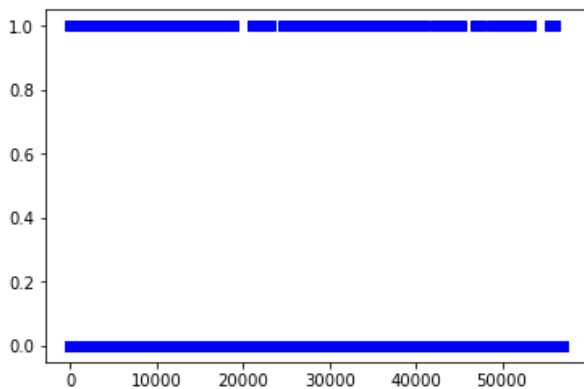


Figura 6: Distribuição Dados Testes mcar_median_*

A Figura 7 e a Tabela VII mostram os resultados para a configuração *muov_median_**. A configuração possui 103 instância contendo fraude. Nesse teste, os algoritmos de aprendizado supervisionado apresentaram desempenho superior aos algoritmos semi-supervisionados.

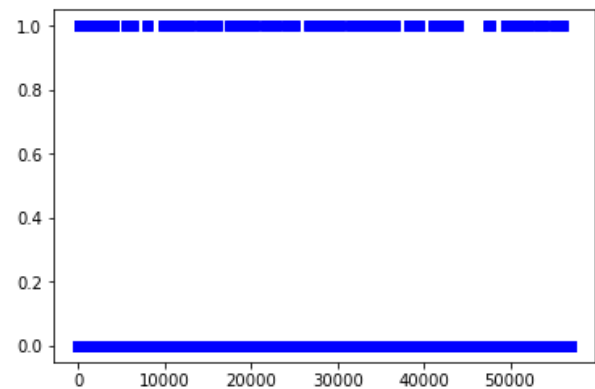


Figura 7: Distribuição Dados Testes muov_median_*

Tabela VII: Resultados muov_median_*

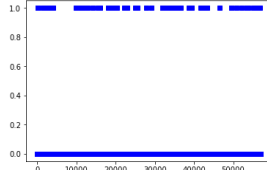
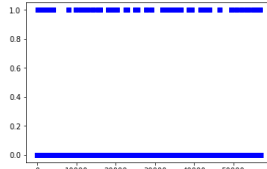
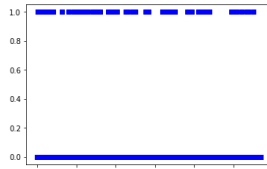
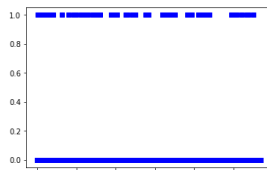
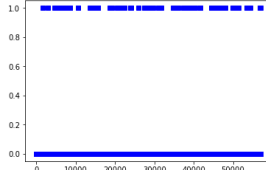
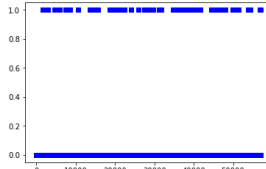
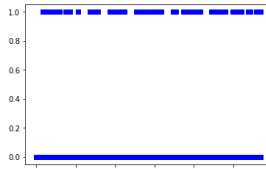
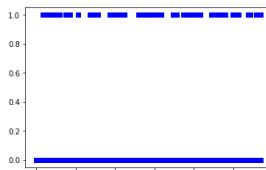
Config	V. Posit	Acurácia	Distribuição
muov_median_svm	65	63,10%	
muov_median_mlp	63	61,16%	
muov_median_ls	62	60,19%	
muov_median_lp	56	54,36%	

Tabela VIII: Resultados miv_median_*

Config	V. Posit	Acurácia	Distribuição
miv_median_svm	70	66,66%	
miv_median_mlp	64	60,95%	
miv_median_ls	70	66,66%	
miv_median_lp	62	59,04%	

A Figura 9 mostra que em geral todos os algoritmos de aprendizado obtiveram médias semelhantes quando se utiliza o método de média, com destaque para o algoritmo *LabelSpreading*, que obteve os melhores resultados em duas configurações, assim como o algoritmo *SVM*.

A Figura 8 apresenta a distribuição para a configuração miv_median_*, onde existem 105 fraudes. A Tabela VIII apresenta os resultados encontrados pelos algoritmos, onde os melhores resultados foram encontrados pelo *SVM* e pelo *MLP*

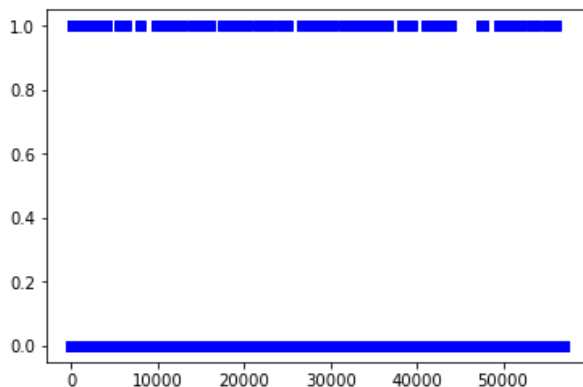


Figura 8: Distribuição Dados Testes miv_median_*

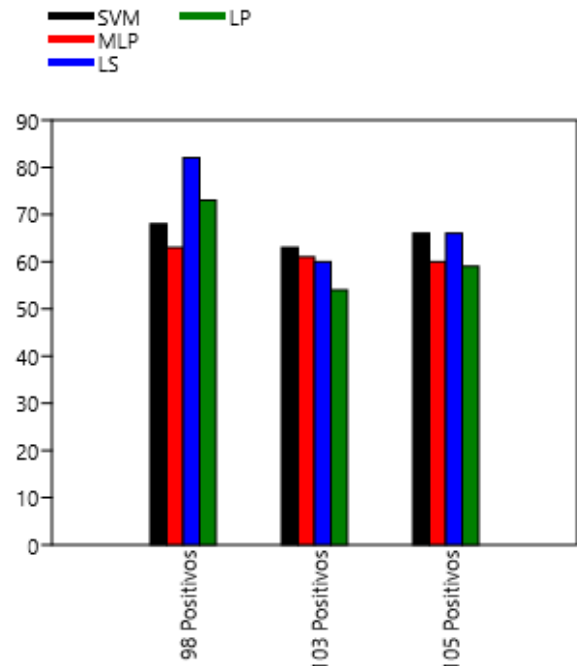


Figura 9: Desempenho *_median_*

C. Método de Substituição *MOST FREQUENT VALUE*

No método de substituição *MOST FREQUENT VALUE* os valores que faltam no *dataset* são substituídos pelo valor que mais se repete dentro do conjunto de valores do atributo correspondente.

Na Figura 10 é possível observar a distribuição para as configurações *mcar_most_**, onde existem 88 fraudes. A Tabela IX mostra que o algoritmo de aprendizado semi supervisionado *LabelSpreading* obteve o melhor resultado novamente.

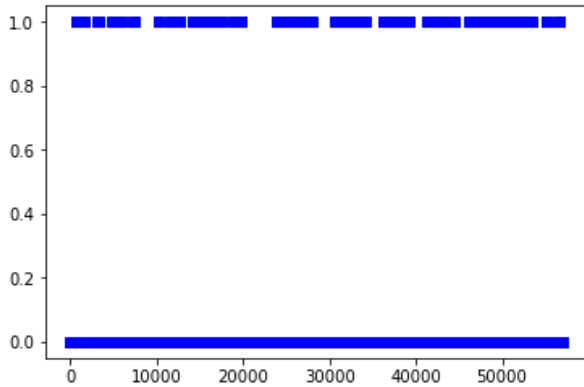


Figura 10: Distribuição Dados Testes *mcar_most_**

Tabela IX: Resultados *mcar_most_**

Config	V. Posit	Acurácia	Distribuição
<i>mcar_most_svm</i>	61	69,31%	
<i>mcar_most_mlp</i>	63	71,59%	
<i>mcar_most_ls</i>	66	75,00%	
<i>mcar_most_lp</i>	58	65,90%	

A Figura 11 apresenta a distribuição para a configuração *muov_most_**, possuindo 106 fraudes. A Tabela X mostra que os algoritmos de aprendizado semi-supervisionado obtiveram os melhores resultados, com vantagem para o *LabelSpreading*.

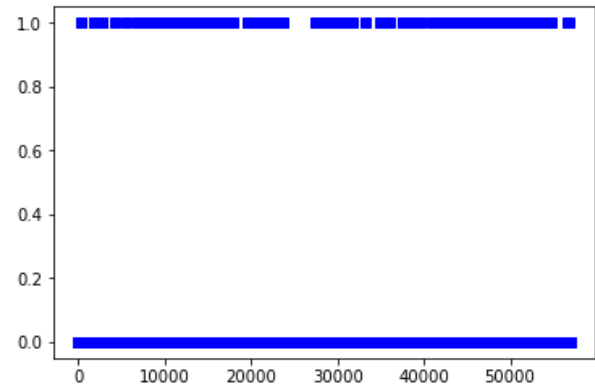


Figura 11: Distribuição Dados Testes *muov_most_**

Tabela X: Resultados *muov_most_**

Config	V. Posit	Acurácia	Distribuição
<i>muov_most_svm</i>	63	59,43%	
<i>muov_most_mlp</i>	66	62,26%	
<i>muov_most_ls</i>	74	69,81%	
<i>muov_most_lp</i>	69	65,09%	

Na Figura 12 é exibida a distribuição para a configuração de testes *miv_most_**, onde temos 113 fraudes. A Tabela XI mostra que os algoritmos semi-supervisionados obtiveram os melhores resultados, com destaque novamente para o *LabelSpreading*.

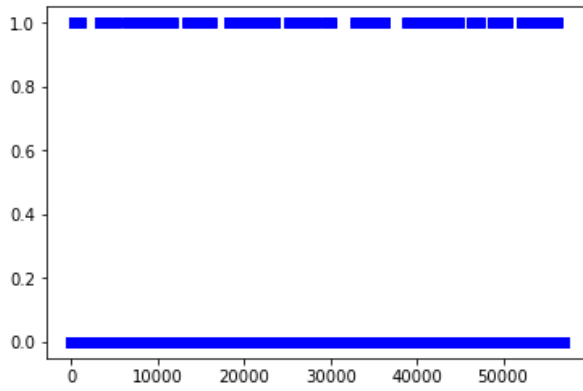


Figura 12: Distribuição Dados Testes miv_most_*

Tabela XI: Resultados miv_most_*

Config	V. Posit	Acurácia	Distribuição
miv_most_svm	63	55,75%	
miv_most_mlp	62	54,86%	
miv_most_ls	82	72,56%	
miv_most_lp	76	67,25%	

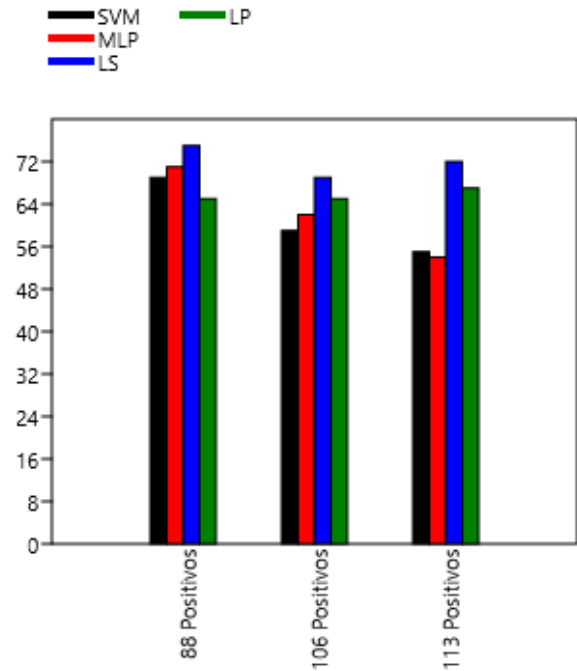


Figura 13: Desempenho *_most_*

VI. CONCLUSÕES

Dados os resultados obtidos na seção anterior, podemos concluir que:

- O algoritmo de aprendizado semi-supervisionado *LabelSpreading* possui o melhor desempenho dentre todos os algoritmos testados.
- O algoritmo de aprendizado semi-supervisionado *LabelPropagation* apresenta desempenho mais sensível em relação a distribuição de amostras de cada classe.
- O algoritmo *LabelSpreading* consegue identificar instâncias em regiões de maior dificuldade no espaço de busca.
- Os métodos de substituição funcionam melhor para *MCAR* na maioria dos casos.

O uso da matriz laplaciana, já citada em IV se mostrou fator essencial quando se trata da diferença entre os algoritmos *LabelSpreading* e *LabelPropagation*. A técnica é utilizada ao se atualizar os pesos das arestas do grafo de similaridade construída no algoritmo que obteve melhor desempenho, possibilitando melhor generalização das características das instâncias que representam uma fraude.

A Figura 13 mostra com clareza que os algoritmos de aprendizado semi-supervisionado obtiveram melhor desempenho, com destaque para o algoritmo *LabelSpreading* que obteve melhores números em todas as configurações.

REFERÊNCIAS

- [1] Aite. 2016 global consumer card fraud: Where card frauds coming from. Technical report, institution, 2016.
- [2] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 159–166. IEEE, 2015.
- [3] Unai Garciarena and Roberto Santana. An extensive analysis of the interaction between missing data types, imputation methods, and supervised classifiers. *Expert Systems with Applications*, 89:52–65, 2017.
- [4] Jiong-Sheng Li and Xiao-Dong Zhang. On the laplacian eigenvalues of a graph. *Linear algebra and its applications*, 285(1-3):305–307, 1998.
- [5] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2014.
- [6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [7] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. Credit card fraud detection using hidden markov model. *IEEE Transactions on dependable and secure computing*, 5(1):37–48, 2008.
- [8] Amarnag Subramanya and Partha Pratim Talukdar. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014.
- [9] Verizon. 2016 data breach investigations report. Technical report, institution, 2016.
- [10] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.

APÊNDICE

A. Principais Funções

```
def _mcarGenerating(data):
    x = _numObservations(data)
    y = _numVariables(data)

    for i in range(int(x * MDP)):
        pos1 = _random([0, x - 1])
        pos2 = _random([0, y - 1])
        data[pos1, pos2] = np.nan
    return data
```

```
def _muovGenerating(data):
    x = _numObservations(data)
    y = _numVariables(data)
    MDVariables = _random([0, y - 1], NV)
    observations = []

    for i in range(int((x * MDP) / NV)):
        observations.append(_random([0, x - 1]))

    for i in range(0, len(observations)):
        for j in range(0, len(MDVariables)):
            data[observations[i], MDVariables[j]] = np.nan

    return data
```

```
def _mivGenerating(data):
    x = _numObservations(data)
    y = _numVariables(data)
    causatives = _random([0, y - 1], NV)
```

```
for i in range(0, len(causatives)):
    observations = []
    aux = data[:, causatives[i]]
    for j in range(0, int((x * MDP) / NV)):
        observations.append(_minIndex(aux))
        aux[observations[j]] = float('inf')
    for j in range(0, len(observations)):
        data[observations[j], causatives[i]] =
            np.nan
    return data
```

```
def _imputationMean(data):
    means = []
    numCols = len(data[0])
    for x in range(0, numCols):
        col = data[:, x]
        means.append(np.mean(col[~np.isnan(col)]))
        col[np.isnan(col)] = means[x]
```

```
def _imputationMedian(data):
    numCols = len(data[0])
    for x in range(0, numCols):
        col = data[:, x]
        sortedCol = np.sort(col)
        median = sortedCol[int(len(sortedCol)
                                / 2)]
        col[np.isnan(col)] = median
```

```
def _imputationMostFrequentValue(data):
    numCols = len(data[0])
    for x in range(0, numCols):
        col = data[:, x]
        unique, pos =
            np.unique(col, return_inverse=True)
        howMany = np.bincount(pos)
        mostFrequent = col[howMany.argmax()]
        col[np.isnan(col)] = float(mostFrequent)
```

```
def simpleNormalization(data):
    numCols = len(data[0])
    for x in range(0, numCols):
        col = data[:, x]
        lvMax = max(col)
        lvMin = min(col)
        for y in range(0, len(col)):
            data[y, x] = (data[y, x] - lvMin) /
                (lvMax - lvMin)
```