Modelos de classificação aplicados a previsão de renda

Professor: Rodrigo Targino

Aluno: Caio Lins

12 de junho de 2022

Conteúdo

Resumo

Neste trabalho, utilizamos classificadores implementados na biblioteca *Sci-Kit Learn*[3] para analisar alguns dados extraídos do censo populacional americano de 1994[1], com o objetivo de prever e explicar a variável resposta, que indica se a renda anual de um indivíduo é maior ou menor que $50000. Após extensa validação cruzada e aplicação de técnicas para contornar a natureza desbalanceada do conjunto de dados, concluímos que a educação, a idade, as relações conjugais e os ganhos capitais externos de cada indivíduo são as variáveis que melhor explicam o *target*. Todo o código utilizado (inclusive o que gerou este documento) pode ser encontrado neste repositório.

1 *Recap* do conjunto de dados

Relembrando o que foi apresentado no relatório anterior, estamos trabalhando com o *"Census Income Data Set"* [1]. Nossa tarefa é classificar cada indivíduo que participou do censo com relação à sua renda: se ele ganha mais que $ 50 000 por ano, ou não. À nossa disposição, temos uma série de variáveis de caráter socioeconômico, categóricas e quantitativas, dentre as quais gostaríamos de identificar aquelas que melhor explicam a variável resposta. Para tanto, treinamos, no nosso conjunto de dados, modelos de classificação que possuem uma boa explicabilidade, ou seja, que não são modelos do tipo "caixa preta". Foram eles, em ordem crescente de complexidade: *perceptron*, regressão logística, modelos baseados em árvores (uma árvore, *bagging* e *random forest*) e *support vector machines (SVM)*.

1.1 *Old problems and new*

No último relatório foi apontado que o conjunto de dados apresentava algumas lacunas, em três *features* diferentes. Apesar de, inicialmente, termos considerado utilizar técnicas estatísticas para preencher os dados, como, por exemplo, o algoritmo EM, por uma questão de (falta de) recursos humanos não foi possível implementar essa ideia. A solução utilizada foi a mais simples: descartamos as amostras que possuíam alguma *feature* faltante. Como elas eram poucas ($\sim$ 2800, contra as $\sim$ 48000 totais), isso não representou uma perda muito significativa.

Entretanto, uma questão não considerada anteriormente se apresentou quando começamos as treinar modelos: o **desbalanceamento do *dataset***. A proporção entre instâncias positivas (ganham mais que 50K) e negativas é de, aproximadamente, 1:3, ou seja, $\sim$ 25% dos indivíduos tem renda elevada. Apesar de não ser exageradamente expressiva, essa assimetria faz com que haja uma tendência para classificar uma determinada instância como negativa.

Ressaltamos que isso não necessariamente é um problema, pois tudo depende de qual é o objetivo da análise que está sendo conduzida. Se a proporção entre classes fora do *dataset* é a mesma de dentro *dataset* e, por algum motivo, deseja-se apenas obter o modelo com melhor *score* [1], não importa muito se há um forte viés para alguma das classes. Entretanto, raramente esse é o caso em aplicações reais de *machine learning*. No nosso problema, por exemplo, não é útil que todos os coeficientes da regressão logística sejam muito baixos, pois isso diz pouco sobre a relevância de cada um para explicar a variável resposta. Portanto, foi necessário utilizar uma estratégia para contrabalancear o viés no conjunto de dados.

2 Metodologia utilizada

O problema de ter um *dataset* desbalanceado é muito comum, e portanto, existem algumas formas já consolidadas de contorná-lo. Duas dessas técnicas são o *over sampling* e *under sampling*. Na primeira, acrescenta-se, artificialemente, mais instâncias da classe menos respresentada ao *dataset*. Isso pode ser feito repetindo instâncias já conhecidas, ou criando instâncias sintéticas. Na segunda, remove-se instâncias pertencentes a classes dominantes. Apesar de bastante

[1] Proporção de instâncias classificadas corretamente.

comumns para resolver esse tipo de problema, essas duas alternativas não foram utilizadas.

A estratégia que utilizamos para contrapor o viés do conjunto de dados envolveu duas components. Primeiramente, selecionamos os hiperparâmetros de cada modelo com um *grid search cross validation*, tomando como referência uma métrica diferente do *score*, que levasse em conta o desempenho dele nas duas classes. As métricas escolhidas foram a **AUC** (*Area Under Curve*), que mede a área debaixo da curva **ROC** (*Receiver Operating Characteristic*), e, no caso do SVM (que não estima probabilidades), a *balanced accuracy score*, que corresponde à média aritmética entre o *recall* [2] e a especificidade [3]. Utilizamos a média aritmética por não termos motivações reais para priorizar o *recall* ou a especificidade.

Em segundo lugar, durante o treino de cada modelo, utilizamos pesos diferentes para as duas classes. Essa abordagem já está implementada nos estimadores do *Sci-Kit Learn* (ao especificar o hiperparâmetro `class_weights = "balanced"`) e corresponde a multiplicar as perdas individuais por um peso específico da classe verdadeira da instância. Para exemplificar essa abordagem e como o cálculo dos pesos é feito, utilizamos [2, Páginas 144-5] como referência, o mesmo artigo utilizado como base pelos desenvolvedores do *Sci Kit Learn* para implementar essa funcionalidade, como pode-se ver aqui.

Suponha que estamos realizando uma regressão logística. Denotamos por π_i a probabilidade estimada pelo modelo de que a i-ésima instância de treino seja positiva, por β os pesos do modelo e por y_i a variável resposta. Então a log-verossimilhança dos dados é dada por

$$\log L(\beta \mid \boldsymbol{y}) = \sum_{y_i=1} \log \pi_i + \sum_{y_i-0} \log(1-\pi_i).$$

Supondo que a proporção real de ocorrência do evento estudado é $\tau \in [0,1]$ e a proporção no conjunto de dados é $\overline{y}$, a log-verossimilhança ponderada pelos pesos $w = \{w_0, w_1\}$ é dada por

$$\log_w L(\beta \mid \boldsymbol{y}) = w_1 \sum_{y_i=1} \log \pi_i + w_0 \sum_{y_i=0} \log(1-\pi_i),$$

onde $w_1 = \tau/\overline{y}$ e $w_0 = (1-\tau)/(1-\overline{y})$. Ao dividir pela proporção no *dataset* e multiplicar pela proporção real, estamos corrigindo o viés provocado pela amostragem. A extensão para múltiplas classes é feita de maneira análoga. Outro link que pode ser de ajuda para entender a ideia do balanceamento é este.

Na implementação do *Sci-Kit Learn*, como é possível ver no link já apresentado, pressupõe-se que, fora do *dataset*, todas as classes possuem a mesma frequência, o que podemos pressupor, com segurança, não ser verdade no nosso caso. Entretanto, ao realizar validação cruzada, vemos que, para **quase todos os modelos testados**, a versão com pesos se saiu melhor do que a versão sem pesos.

2.1 Abordagem geral

Feita essa discussão sobre dados desbalanceados, especificiamos agora qual foi a abordagem geral para treinar os modelos. Inicialmente, separamos 25% dos dados para teste e o restante para treino. O dado de teste só foi utilizado para computar os resultados apresentados na

[2] *Score* dentro de uma dada classe. Em classificação binária, se refere ao *score* para a classe positiva.
[3] *Recall* para a classe negativa.

seção seguinte. Em seguida, como mencionamos anteriormente, selecionamos os melhores hiperparâmetros para cada modelo com um *grid search cross validation*, por meio de uma função disponível no *Sci-Kit Learn*. Procuramos explorar ao máximo o espaço de hiperparâmetros, dados os recursos computacionais disponíveis.

3 Resultados dos Modelos

3.1 Perceptron

O perceptron é o modelo linear mais simples, pois apenas realiza uma combinação linear das *features* e classifica o indivíduo de acordo com o sinal do resultado. O melhor modelo encontrado utiliza regularização do tipo *elastic-net*, com peso para regularização de Lasso igual a 0.3. O peso para a regularização como um todo foi 0.001.

Na Figura 1 podemos ver a matriz de confusão e a curva ROC para este modelo. As principais métricas do modelo foram:

- *Recall*: 90.7%.
- Precisão: 32.8%.
- Especificidade: 38.8%.
- AUC: 0.648 (0.852 no treino).

Para que os coeficientes tivessem valores significativos, o perceptron foi treinado com dados devidamente padronizados, para terem média zero e variância 1. As *features* com maiores pesos estão representadas na Figura 2. Podemos ver claramente que, de acordo com o perceptron, ter um cônjuge nas forças aramadas (`af` significa *armed forces*) contribui de forma positiva para ter uma renda alta, enquanto ser do Vietnã, de forma negativa. O campo `capital_gain` não estava muito bem explicado na fonte do dataset, porém é razoável assumir que se trata dos ganhos monetários de fontes além do salário, como investimentos, por exemplo.

3.2 Regressão Logística

A Regressão Logística já é um modelo um pouco mais sofisticado, por usar uma ativação não linear. Também utilizamos um *Scaler* para padronizar os dados antes de cada chamada ao método *fit*, de modo que os coeficientes possam ser utilizados para medir a relevância das *features*. O melhor modelo encontrado utiliza regularização do tipo *elastic-net*, com peso para regularização de Lasso igual a 0.25. A forma como o peso é dado para a regularazição como um todo é diferente do perceptron. O termo da perda correspondente aos dados é multiplicado por uma constante C, que, no caso do modelo validado, foi igual a 0.01, o que significa uma regularização bastante elevada.

Na Figura 3 podemos ver a matriz de confusão e a curva ROC para este modelo. As principais métricas do modelo foram:

- *Recall*: 91.5%.
- Precisão: 50.8%.

- Especificidade: 70.7%.
- AUC: 0.901 (0.904 no treino).

As *features* com maiores pesos estão representadas na Figura 4. Por ela, concluímos que ter elevados ganhos de renda além do salário (`capital_gain`), bem como ter uma boa educação, são fatores que contribuem para uma ter uma elevada renda. Por outro lado, nunca ter se casado (`marriage_never_married`) contribui negativamente.

3.3 Modelos baseados em árvores

Para os modelos baseados em árvores, uma desvantagem comparativa foi o fato de que a implementação de árvores do *Sci-Kit Learn* não suporta *features* binárias, como é possível ver no último parágrafo desta seção (1.10.6) do *user guide* do *Sci-Kit Learn*. Para confirmar isso, ao tentarmos treinar com variáveis categóricas, geramos uma falha, pois o algoritmo esparava apenas floats.

3.3.1 Uma árvore

Com o intuito de começar simples e depois complicar, primeiro treinamos uma única árvore. Os melhores parâmetros escolhidos por validação cruzada foram:

- Um α relativo à poda de custo-complexidade igual a 0.001.
- Profundidade máxima igual a 9.
- Um mínimo de 1% de samples em cada folha.

Na Figura 5 podemos ver a matriz de confusão e a curva ROC para este modelo. As principais métricas do modelo foram:

- *Recall*: 49.9%.
- Precisão: 70.2%.
- Especificidade: 93.0%.
- AUC: 0.84 (0.844 no treino).

As *features* com mais importância estão retratadas na Figura 6. Essa importância é calculada como sendo a redução total da perda, normalizada, causada por aquela *feature*. Podemos ver que, de acordo com uma única árvore, o grau de educação de um indivíduo e a sua idade são duas boas variáveis para explicar a sua renda. Por se tratar de apenas *uma* árvore, podemos representá-la graficamente, o que é feito na Figura 7

3.3.2 *Bag of trees*

O próximo passo é o de ajustar um *bagging* classifier nos dados, utilizando árvores como classificadores básicos. Esse tipo de estimdor tem como objetivo reduzir a variância do modelo, de modo que se espera que ele generalize melhor. Por uma questão de falta de recursos computacionais, como parâmetros para cada classificador dentro do *bag*, utilizamos os melhores parâmetros encontrados por validação cruzada para uma única árvore. Com relação ao número

de estimadores, vemos que, como esperado, quanto mais estimadores utilizamos, melhor é o desempenho do modelo. Entretanto, a partir de cerca de 100 estimadores, o ganho marginal começou a ser muito pequeno.

Na Figura 8 podemos ver a matriz de confusão e a curva ROC para este modelo. As principais métricas do modelo foram:

- *Recall*: 27.4%.
- Precisão: 89.1%.
- Especificidade: 99%.
- AUC: 0.856 (0.862 no treino).

Vemos que, apesar do *recall* ter caído, todas as outras métricas subiram.

3.3.3 *Random Forest*

O *random forest* é um *bag*, porém com uma tentativa de descorrelacionar as árvores. Todas as considerações feitas para o *bag of trees* se aplicam aqui.

Na Figura 9 podemos ver a matriz de confusão e a curva ROC para este modelo. As principais métricas do modelo foram:

- *Recall*: 46.9%.
- Precisão: 72.27%.
- Especificidade: 94.06%.
- AUC: 0.855 (0.859 no treino).

Podemos ver que, apesar de uma queda na precisão e na especificidade com relação ao modelo de *bagging*, houve um aumento do *recall*, de modo que podemos concluir que o modelo ficou mais balanceado.

Para o *random forest*, novamente temos uma forma prática de calcular as importâncias de cada *feature*, já disponível como um método no *Sci-Kit Learn*. Ela é exatamente a mesma do classificador com apenas uma árvore. As *features* mais importamentes de acordo com essa métrica estão representadas na Figura 10. Vemos que as conclusões são exatamente as mesmas obtidas quando consideramos apenas uma árvore.

3.4 SVM

As *Support Vector Machines* são a classe de modelos mais complexa que consideramos. Utilizamos um SVC (*Support Vector classifier*) em conjunção com um *scaler*. Fizemos um *grid search* para encontrar o melhor *kernel* e parâmetro de regularização. O resultado obtido foi um *kernel* linear, com parâmetro de regularização $C = 1$ (veja a seção sobre regressão logística). Como o SVM não gera probabilidades, a validação cruzada não foi realizada com a métrica AUC, e sim com a *balanced accuracy score*, que, como dito na seção anterior, é a média entre o *recall* e a especificidade. As principais métricas do modelo foram:

- *Recall*: 85.55%.

- Precisão: 57.70%.
- Especificidade: 77.59%.
- *balanced accuracy score*: 0.816 (0.813 no treino).

Podemos ver que o modelo se saiu melhor no teste do que no treino. Também vemos que ele apresenta um conjunto de métricas muito bom, comparado aos outros classificadores, por ser o único que conseguiu tanto *recall* quanto especificidade elevados. Na Figura 11 podemos ver a matriz de confusão para este modelo.

4 Conclusão

Neste trabalho, nos propusemos a utilizar modelos de classificação para estudar o conjunto de dados [1], com objetivo de prever/explicar a variável resposta, relativa à renda dos indivíduos.

Nesse sentido, na primeira parte realizamos uma análise exploratória, idenficando variáveis de interesse e possíveis problemas que deveríamos contornar. Em primeiro lugar, nos deparamos com dados faltantes. A solução utilizada para esse problema foi a de simplesmente desconsiderar os indivíduos que possuíssem alguma *feature* latente. Em segundo lugar, percebemos que o desbalanceamento do *dataset* poderia criar um viés indesejado em favor da classe negativa. Para contrabalancear essa tendência, utilizamos um mecanismo de pesos para cada classe, que demonstrou ser efetivo em quase todos os modelos analisados.

Ao treinarmos os classificadores, vimos que a estratégia de validação cruzada utilizada foi efetiva, pois as perdas nos conjuntos de treino e teste ficaram muito próximas. Isto é, não houve overfitting significativo. Também vimos que as *features* que mais foram classificadas como relevantes foram: `capital_gain`, o grau de educação, a idade e as relações conjugais de cada indivíduo. A *feature* `capital_gain` é a única cujo significado não é inteiramente conhecido, por não ser informado na página do *dataset*. Por ser diferente de `target`, podemos inferir que se trata de ganhos monetários além de salários.

Referências

[1] Dheeru Dua e Casey Graff. *UCI Machine Learning Repository – Census Income Data Set*. 2017. URL: `https://archive.ics.uci.edu/ml/datasets/Census+Income`.

[2] Gary King e Langhe Zeng. «Logistic Regression in Rare Events Data». Em: *Society for Political Methodology* (2001), pp. 137–163.

[3] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». Em: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

Lista de Figuras

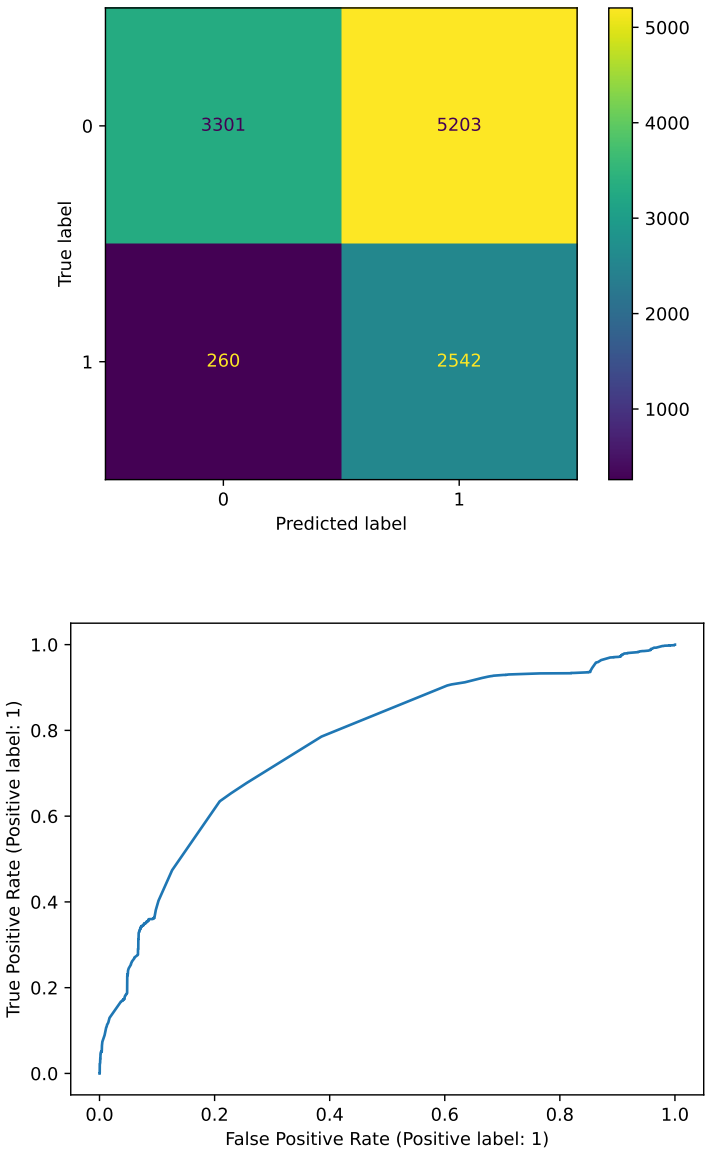

Figura 1: Matriz de confusão e curva ROC para o modelo perceptron. AUC = 0.648.

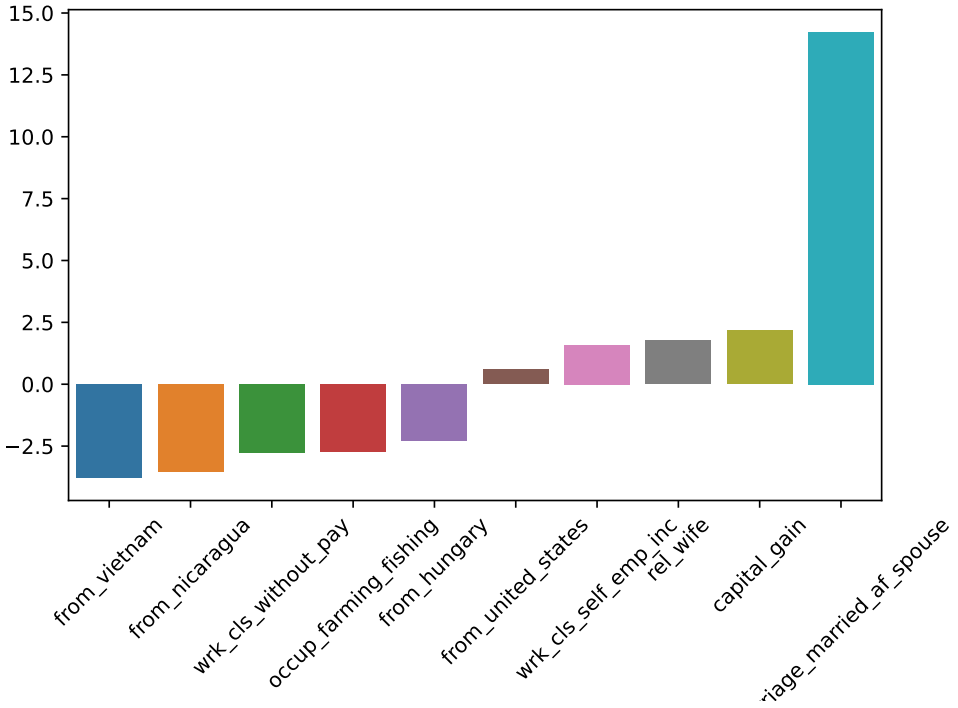

Figura 2: Feautures com maiores valores absolutos de pesos, positivos e negativos, para o perceptron.

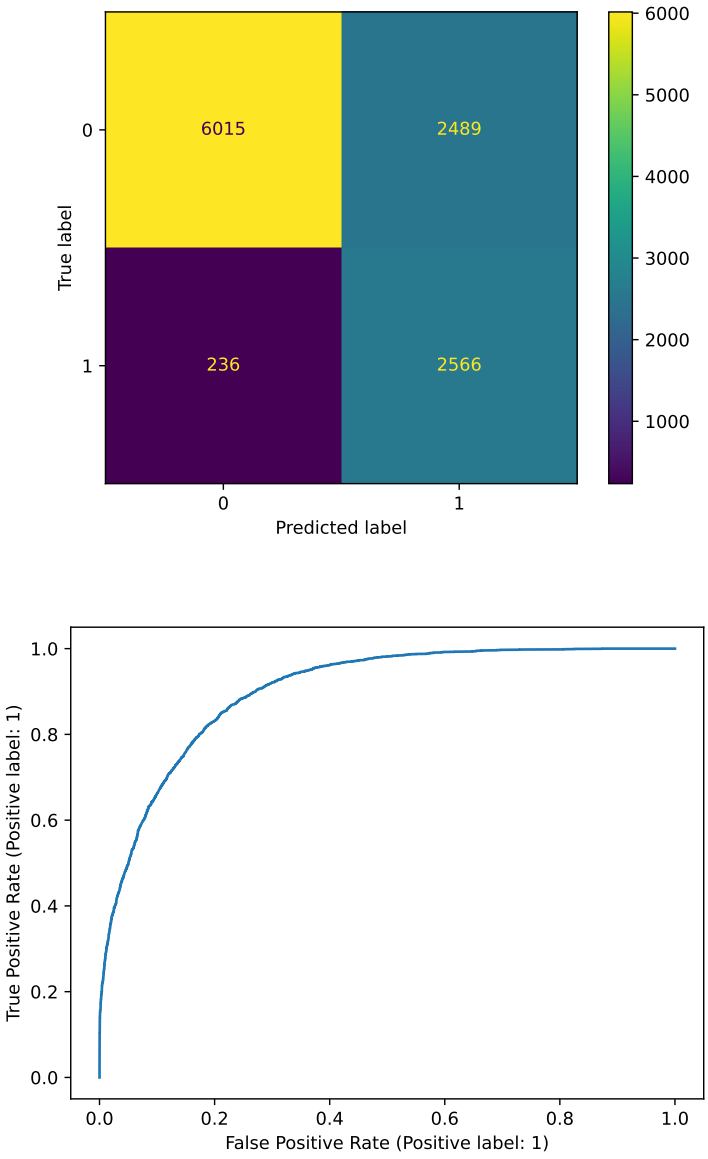

Figura 3: Matriz de confusão e curva ROC para o modelo de regressão logística. AUC = 0.901.

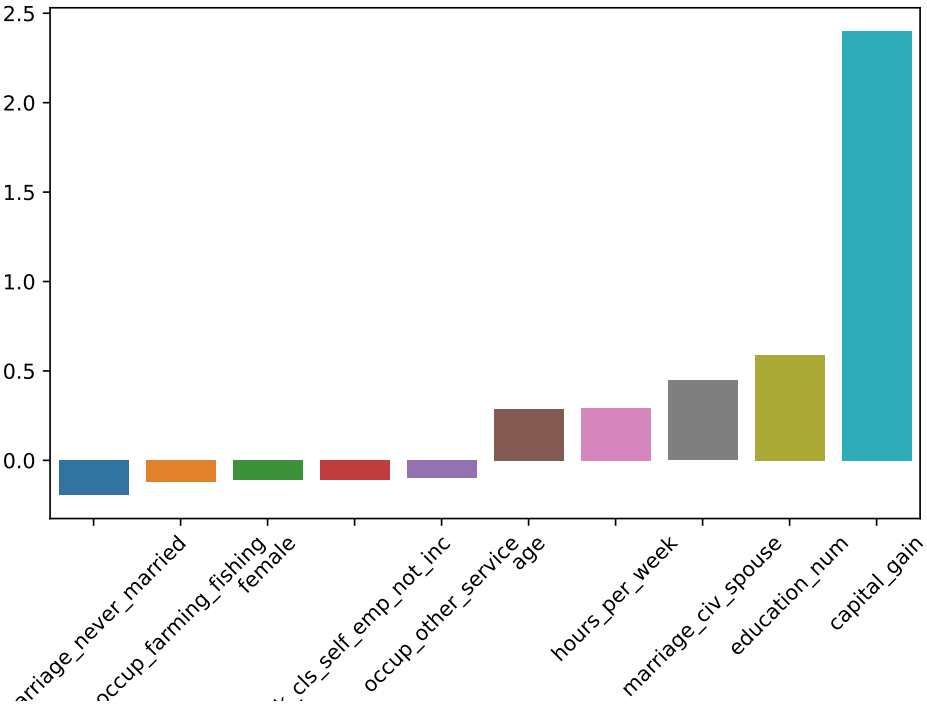

Figura 4: Feautures com maiores valores absolutos de pesos pesos, positivos e negativos, para a regressão logística.

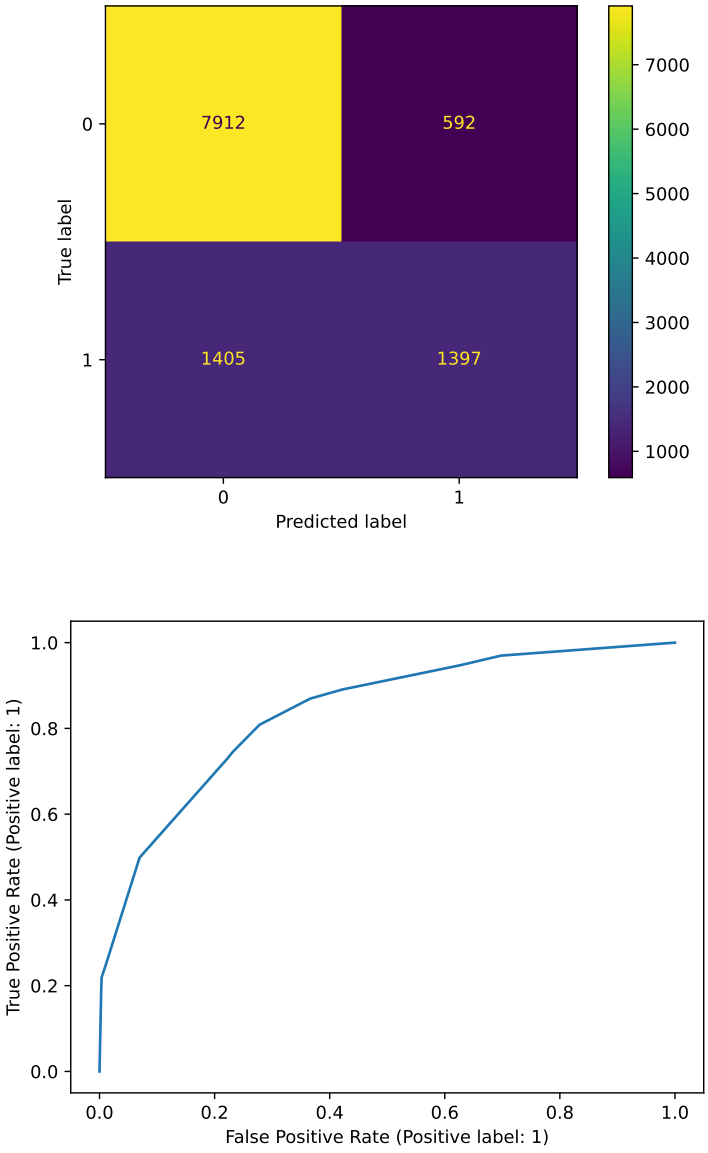

Figura 5: Matriz de confusão e curva ROC para uma única árvore. AUC = 0.838.

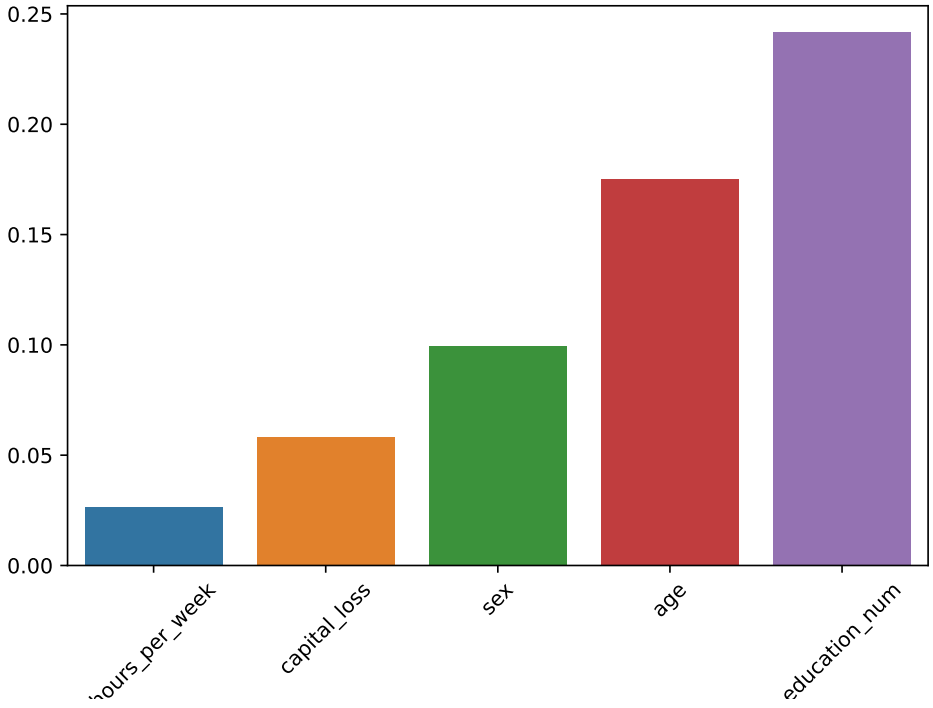

Figura 6: Features com maiores importâncias, de acordo com uma única árvore.

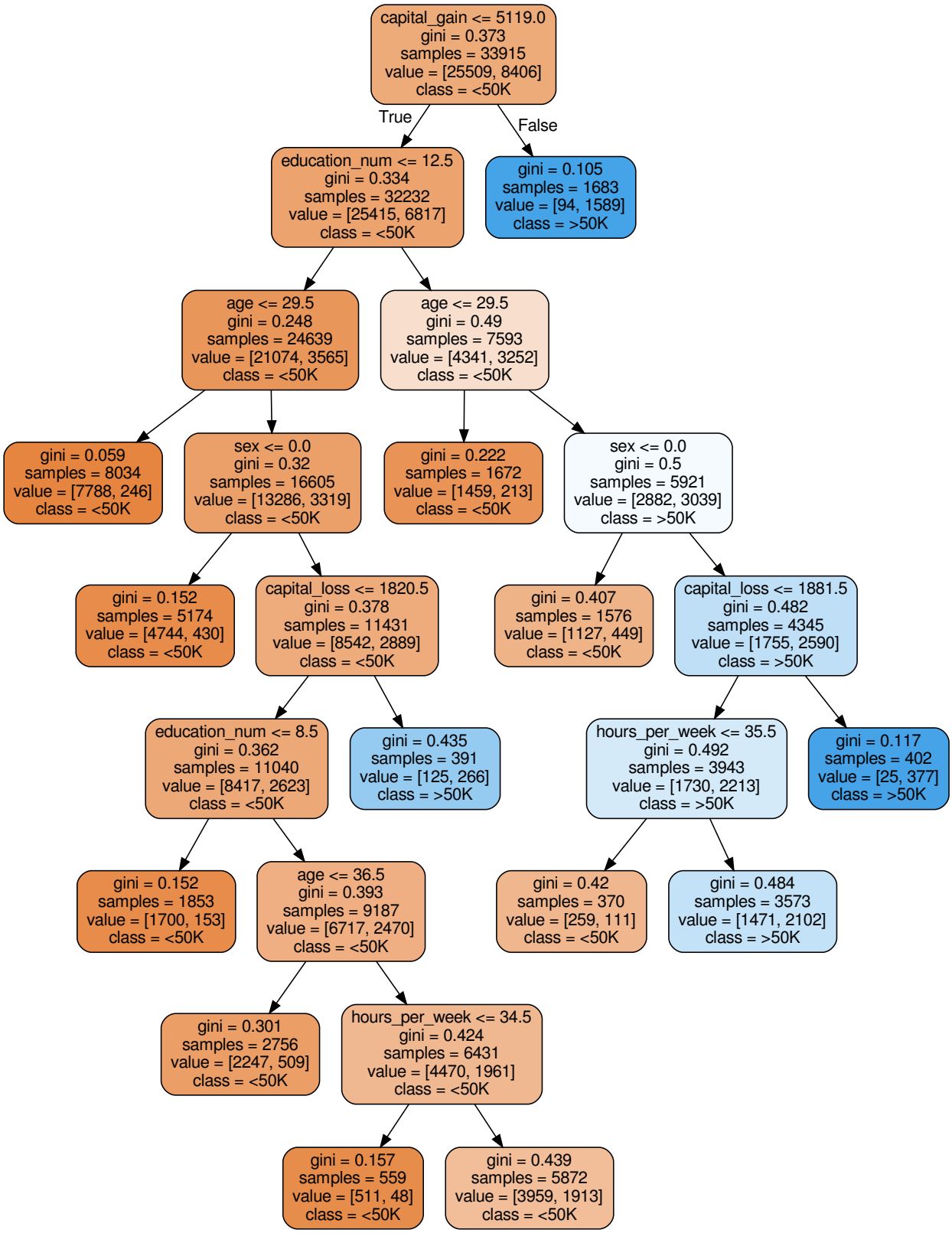

Figura 7: Árvore obtidas ao fitarmos apenas um classificador no dataset.

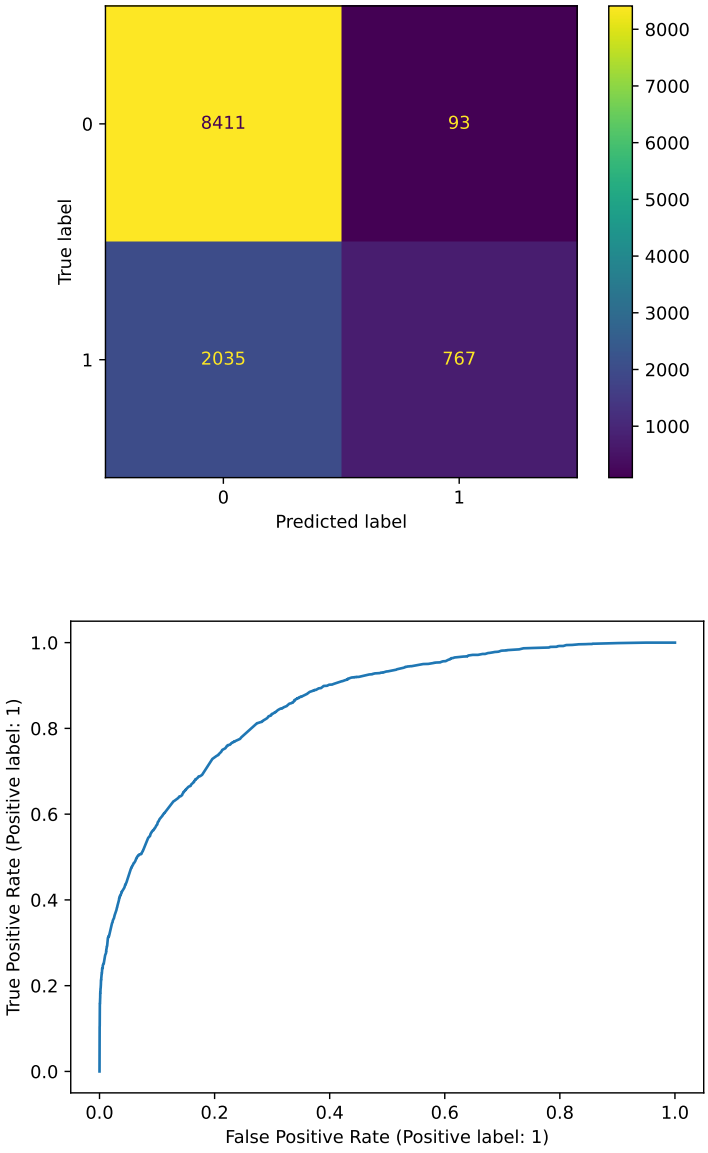

Figura 8: Matriz de confusão e curva ROC para um *bag of trees*. AUC = 0.856.

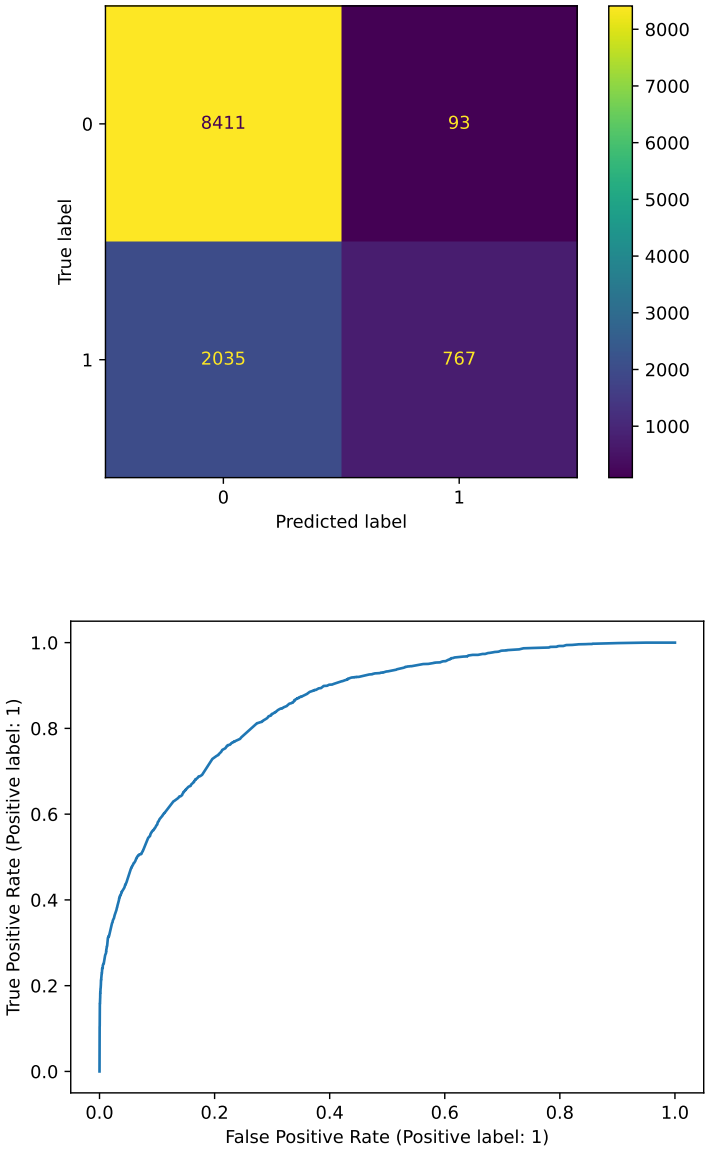

Figura 9: Matriz de confusão e curva ROC para um *bag of trees*. AUC = 0.856.

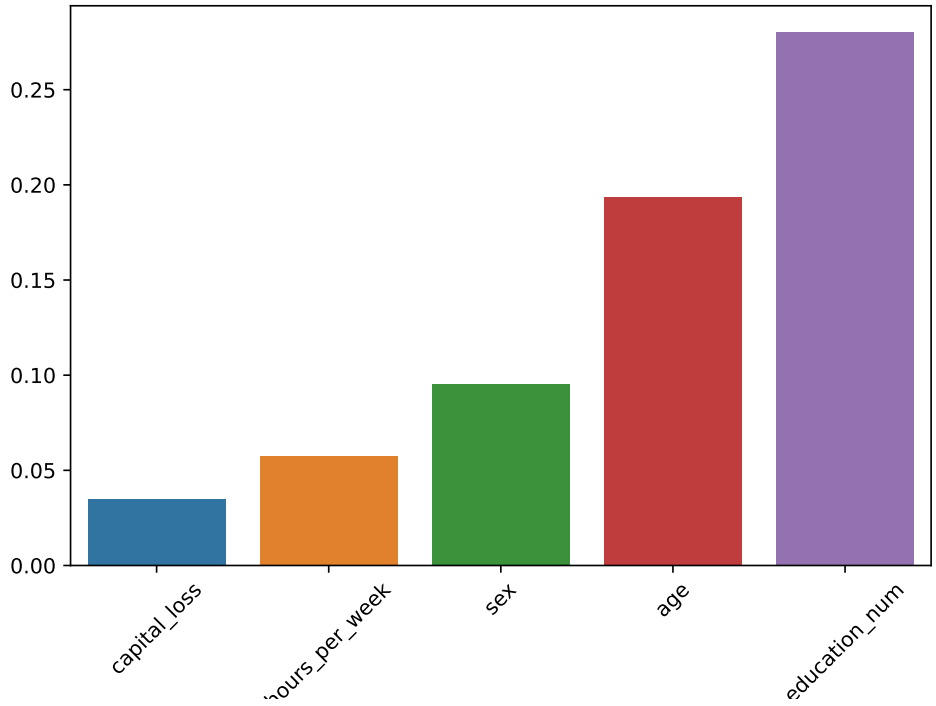

Figura 10: Features com maiores importâncias, de acordo com uma *random forest*.

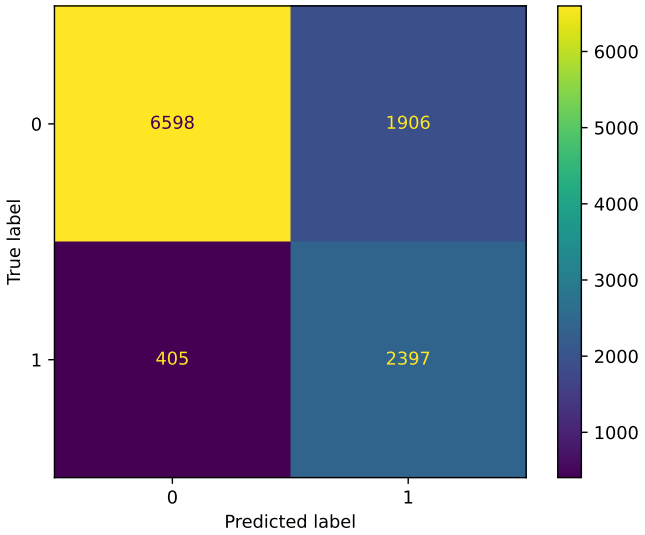

Figura 11: Matriz de confusão para o SVM.