Universidade São Judas Tadeu

Arthur Lisboa Jacom
Gustavo Lippi Eugenio
Isabella Grisma da Silva Braz
Matheus Sacho Aga
Gabriel Lins
Eduardo Chiaratto

Relatório Final
Comitê de Classificadores

,
Professora Gabriela Oliveira Biondi

São Paulo
2020

Sumário

1. Objetivo do Projeto
2. Base de dados Census Adult Income
3. Várivavel-target Sex
4. Explorando a base de dados
5. Presença dos dados
6. Pré-processamento de dados
    a. Transformar dados numéricos em strings
    b. Transformar strings em dados binários

7. Método de Avaliação
8. Algoritmos
    a. Árvore de Decisões
    b. Regressão Logística
    c. KNN
    d. Naive Bayes
    e. Redes Neurais

9. Resultados

Objetivo do projeto

O objetivo desse projeto é encontrar um algoritmo que seja capaz de prever o sexo dos indivíduos com base em outras informações.

Para isso iremos utilizar a base de dados Census que conta com 14 tipos de dados pessoais de mais de 30 mil pessoas e com esses dados iremos construir um comitê de classificadores para descobrir qual algoritmo prever com maior precisão.

Base de dados Census Adult Income

A base de dados Census que vamos utilizar foi criada em 1994 pelo United States Census Bureau (USCB), a principal agência do sistema de estatística federal dos EUA, ela faz parte do departamento de comércio e o seu diretor é escolhido pelo presidente dos Estados Unidos.

Seu propósito é coletar dados das pessoas e da economia norte-americana a cada 10 anos a fim de ajudar o estado, as comunidades locais e os comércios a tomarem decisões mais assertivas.

Váriavél-target Sex

Naquela época os papéis do homem e da mulher eram claramente divididos em padrões que limitavam nossa percepção da capacidade do individuo pelo seu sexo.

Nossa variável-target será a coluna Sex e o desafio dos nossos algoritmos será utilizar os dados disponíveis para encontrar esses padrões e classificar os indivíduos como homens ou mulheres.

Apesar de hoje em dia muita coisa já ter mudado, acreditamos que ainda há muito a ser feito, e visualizar esses padrões com os algoritmos pode nos ajudar a encontrar soluções mais eficazes para combater a desigualdade social.

Explorando a base de dados

A base de dados utilizada está disponível para download na internet através do site UCI Machine Learning.

Dentro dela existem 14 colunas ou *Features* com diferentes dados, abaixo temos um resumo sobre o que cada dado significa.

1. **age** = A idade representada por números inteiros.
   Teenager (17-19), Young (20-29), Adult (30-55), Senior (56-70) and Old (71+).

2. **workclass** = Classe de trabalho.
   Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay ou Never-worked.

3. **final.weight** = O termo estimativa se refere aos totais populacionais derivados do CPS pela criação de "contagens ponderadas" de quaisquer características socioeconômicas específicas da população. Para saber mais, clique aqui.
   Números inteiros.

4. **education** = Nível de estudo.
   Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

5. **education.num** = Quantos anos que o cidadão passou estudando representado por números inteiros.
   Números inteiros.

6. **marital-status**: Estado civil.
   Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

7. **occupation**: Área de atuação da profissão do cidadão.
   Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

8. **relationship**: Tipo de relação com a família.
   Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

9. **race**: Raça.
   White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

10. **sex**: Sexo masculino ou feminino.

Female, Male.

11. **capital-gain** and **capital-loss**: Ganhos ou perdas de capital para um número inteiro individual maior ou igual a 0.
    Números decimais.

12. **hours-per-week**: Quantidade de horas que o cidadão trabalha por semana
    Part-time (0-25), Full-time (26-45), Over-time (46-65) and Too-much (66+).

13. **native-country**: País de origem.
    United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Presença dos dados

Além de entender o que cada dado significa, é importante sabermos qual a porcentagem de espaço que ele ocupa na base para tomarmos decisões importantes durante a etapa de pré-processamento.

Coluna	Entrada	% na Base de Dados

Na coluna workclass é interessante ver que uma porcentagem quase nula executava trabalhos voluntários (without-pay) e que a **grande maioria trabalhava para empresas privadas**.

Workclass	Private	73.88%
	Self-emp-not-inc	8.28%
	Local-gov	6.85%
	State-gov	4.24%
	Self-emp-inc	3.56%
	Federal-gov	3.12%
	Without-pay	0.04%

Mais da metade dos cidadãos haviam se formado no colégio e/ou ingressado em alguma faculdade (HS-grad e Some-college), mas é interessante que apenas 16% deles possuem bacharelado, 5% tiveram mestrado e apenas cerca de 1% fizeram doutorado, isso mostra como o acesso ao ensino superior naquela época era limitado até para um país desenvolvido.

Education	HS-grad	32.62%
	Some-college	22.14%
	Bachelors	16.72%
	Masters	5.39%
	Assoc-voc	4.33%
	11th	3.47%
	Assoc-acdm	3.34%
	10th	2.71%
	7th-8th, Prof-school, 9th, 12th e Doctorate	(x) 1.84% ~ 1.24%
	5th-6th, 1st-4th e Preschool	0.95% ~ 0.14%

Cerca de **78% da população estava casada ou nunca tinha se casado**, aqui podemos observar que o divórcio ainda não era bem visto.

Marital Status	Married-civ-spouse	46.63%
	Never-married	32.24%
	Divorced	13.97%
	Separated	3.11%
	Widowed	2.74%
	Married-spouse-absent	1.22%
	Married-AF-spouse	0.06%

As ocupações eram bem diversificadas, embora algumas fossem mais ou menos populares que outras.

Occupation	Prof-specialty, Craft-repair e Exec-managerial	13.38% ~ 13.23%
	Adm-clerical	12.33%
	Sales	11.88%
	Other-service	10.64%
	Machine-op-inspct	6.51%
	Transport-moving	5.21%
	Handlers-cleaners	4.47%
	Farming-fishing e Tech-support	3.27% ~ 3.02%
	Protective-serv	2.13%
	Priv-house-serv e Armed-Forces	(x) 0.47% ~ 0.02%

É interessante como existem muito mais maridos (Husband) do que esposas (Wife), já que para que o homem seja um marido, ele precisa de uma esposa.

Relationship	Husband	41.32%
	Not-in-family	25.61%
	Own-child	14.80%
	Unmarried	10.64%
	Wife	4.66%
	Other-relative	2.94%

Que a maioria dos norte-americanos são brancos não é novidade nenhuma, mas me assusta pensar que **apenas cerca de 9% eram negros**.

Race	White	85.97%
	Black	9.33%
	Asian-Pac-Islander	2.96%
	Amer-Indian-Eskimo	(x) 0.94%

	Other	(x) 0.76%

Pelo fato de 67.5% das pessoas entrevistadas serem homens, **é mais provável que os algoritmos encontrem padrões mais assertivos para homens** do que para mulheres.

Sex	Male	67.56%
	Female	32.43%

Naturalmente a maioria das pessoas que vivem nos estados unidos são norte americanas, mas **a porcentagem de alguns países chega a ser menor que 0.00**%.

Native Country	United-States	91.18%
	México	2.02%
	Philippines, Germany, Puerto-Rico, Canada, India, El-Salvador, Cuba, England, Jamaica, South, Italy, China, Dominican-Republic, Vietnam, Guatemala, Japan, Poland, Columbia, Taiwan, Haiti, Iran, Portugal, Nicaragua, Peru, Greece, Ecuador, France, Ireland, Hong, Trinadad & Tobago, Cambodia, Thailand, Laos, Yugoslavia, Outlying-US (Guam-USVI-etc), Hungary, Honduras, Scotland, Holand - Netherlands	(x) 0.623% ~ 0.003%

Sabemos que as pessoas com maior poder aquisitivo são sempre minoria na sociedade, entretanto quase **25% dos americanos recebiam mais do que 50 mil por ano**.

	<=50K	75.10%

Income	>50 mil	24.89%

Pré-processamento de dados

Nossa base de dados é composta por colunas com dados do tipo object, que são as colunas com palavras, e com dados int64 que são colunas com dados numéricos contínuos, nosso objetivo nessa etapa será transformar todas essas entradas em dados binários de 0 e 1.

Para isso iremos dividir nosso pré-processamento de dados em duas etapas, primeiro iremos converter os dados numéricos em strings e depois vamos converter as strings em dados binários.

Transformar dados numéricos em strings

Nós iremos transformar os dados numéricos das colunas 'age' e 'hours.per.week' em dados string.

- **Age**: Vamos classificar intervalos de números em diferentes strings, sendo *Teenager* (17-19), *Young* (20-29), *Adult* (30-55), *Senior* (56-70) e *Old* (71+).

```
def fase(idade):
  if idade.iloc[0] <= 19:
    idd = 'Teenager'
  elif idade.iloc[0] <= 29:
    idd = 'Young'
  elif idade.iloc[0] <= 55:
    idd = 'Adult'
  elif idade.iloc[0] < 70:
    idd = 'Senior'
  else:
    idd = 'Old'
  return idd
trn['age'] = trn[['age']].apply(fase, axis=1)
```

```
trn.age

0          Adult
1          Adult
2          Adult
3          Adult
4          Young
           ...
30157      Young
30158      Adult
30159     Senior
30160      Young
30161      Adult
Name: age, Length: 30162, dtype: object
```

- **Hours per week**: Vamos classificar intervalos de números em diferentes strings, sendo *Part-time* (0-25), *Full-time* (26-45), *Over-time* (46-65) e *Workaholic* (66+).

> *Workaholic é uma expressão em inglês que significa "Vício em trabalho".*

A jornada de trabalho padrão no mundo é entre 40 e 44 horas por semana, mas não é assim em todos os lugares, na França por exemplo a jornada semanal é de de 35 horas, enquanto na Coreia do Norte pode chegar até 112 horas por semana.

```
def hours(pwek):
  if pwek.iloc[0] <= 25:
    hpw = 'Part-time'
  elif pwek.iloc[0] <= 45:
    hpw = 'Full-time'
  elif pwek.iloc[0] <= 65:
    hpw = 'Over-time'
  else:
    hpw = 'Workaholic'
  return hpw
trn['hour_per_week'] = trn[['hour_per_week']].apply(hours,axis=1)
```

```
trn.hour_per_week
```

```
0         Full-time
1         Part-time
2         Full-time
3         Full-time
4         Full-time
           ...
30157     Full-time
30158     Full-time
30159     Full-time
30160     Part-time
30161     Full-time
Name: hour_per_week, Length: 30162, dtype: object
```

Transformar strings em dados binários

Para que o resultado do nosso projeto tenha resultados conclusivos precisamos excluir algumas colunas de dados que são pouco ou nada importantes na proposta do projeto.

Vamos excluir a coluna educantion_num pois já vamos utilizar a coluna education que possui basicamente os mesmos dados, Unnamed: 0, capital_gain, capital loos e final_weight não são informações relevantes ao projeto.

```
#Excluir colunas com dados irrelevantes.
trn.drop(['Unnamed: 0', 'final_weight', 'education_num',
'capital_gain', 'capital_loos'], axis=1, inplace=True)
```

```
trn.dtypes
#Converter todas as colunas para dados numéricos
```

```
age               object
workclass         object
education         object
marital_status    object
occupation        object
relationship      object
race              object
sex               object
hour_per_week     object
native_country    object
income            object
dtype: object
```

Depois vamos converter os dados strings em dados binários das colunas restantes.

```
ag = pd.get_dummies(trn['age'], drop_first=False)
wc = pd.get_dummies(trn['workclass'], drop_first=False)
ed = pd.get_dummies(trn['education'], drop_first=False)
ms = pd.get_dummies(trn['marital_status'], drop_first=False)
oc = pd.get_dummies(trn['occupation'], drop_first=False)
rs = pd.get_dummies(trn['relationship'], drop_first=False)
rc = pd.get_dummies(trn['race'], drop_first=False)
sx = pd.get_dummies(trn['sex'], drop_first=True)
hw = pd.get_dummies(trn['hour_per_week'], drop_first=False)
nc = pd.get_dummies(trn['native_country'], drop_first=False)
ic = pd.get_dummies(trn['income'], drop_first=True)
```

Depois vamos remover as colunas originais e juntar as novas colunas binárias à base de dados.

```
trn.drop(['age','workclass','education','marital_status','occupation','
relationship','race','sex', 'hour_per_week','native_country','income'],
axis=1, inplace=True)

trn = pd.concat([trn, ag, sx, rc, rs, wc, oc, ms, nc, hw, ed, ic],
axis=1)
```

Agora que os dados já estão convertidos, vamos aplicar algumas normatizações nos nomes das colunas.

```
#Remover todos os espaços em branco nos nomes das colunas
trn.columns = trn.columns.str.replace(' ', '')
trn.columns = trn.columns.str.replace('-', '_')
trn.columns = trn.columns.str.replace('.', '_')

#Renomear colunas para que a leitura dos resultados fique mais clara.
trn = trn.rename(columns = {'Male': 'Sex', 'Amer_Indian_Eskimo':
'Amer_Indian','Asian_Pac_Islander':'Asian','Outlying_US(Guam_USVI_etc)'
: 'Outlying_US', '>50K': 'Income_>50k'}, inplace = False)
```

Para tentar aumentar o nível de acurácia após as etapas de pré-processamento nós excluímos as colunas que os dados fossem igual ou menor que 1% da base dados, tais como “Race: Others”, “Workclass: Without-pay” e “Education: Preschool”. Entretanto o resultado foi o aumento de 0.10% na acurácia

do nosso modelo, mas para não influenciar os algoritmos a certos resultados resolvemos manter essas colunas.

```
trn.head()
```

	Adult	Old	Senior	Teenager	Young	Sex	Amer_Indian	Asian	Black	Other	White
0	1	0	0	0	0	1	0	0	0	0	1
1	1	0	0	0	0	1	0	0	0	0	1
2	1	0	0	0	0	1	0	0	0	0	1
3	1	0	0	0	0	1	0	0	1	0	0
4	0	0	0	0	1	0	0	0	1	0	0

Agora que todos os dados estão prontos para serem usados no formato binário (0 e 1), vamos separar nossa base entre treino e teste.

```
#Divisão do dataset entre treino e teste
X_train, X_test, y_train, y_test =
train_test_split(trn.drop('Sex',axis=1), trn['Sex'], test_size=0.30,
random_state=100)
```

Nós realizamos um teste onde duplicamos a base de dados para que o algoritmo usasse 100% da base tanto para treino quanto para teste. Quando aplicamos o algoritmo, o resultado é um aumento de cerca de 1% na acurácia em relação a uma divisão de 70% Treino e 30% Teste, então decidimos que seria melhor continuar realizando a divisão do que dobrar a nossa quantidade de dados.

Método de Avaliação

Com a base de dados preparada podemos começar a aplicar e avaliar nossos algoritmos utilizando o classification report para observar a acurácia, que determina a capacidade do nosso modelo de fazer classificações e o F1-Score que traz uma média entre o Precision que avalia os verdadeiros positivos e o Recall, que avalia a capacidade do modelo de encontrar todos os resultados positivos.

Algoritmos

Vamos modelar e avaliar cinco algoritmos diferentes na base de dados, são eles: Árvore de Decisões, Regressão Logística, KNN, Naive Bayes e Redes Neurais.

Árvore de Decisões

O primeiro algoritmo que nós utilizamos foi a Árvore de decisões, que apresentou resultados não satisfatórios, já que usava uma quantidade pequena de amostras para as classificações, o que acarretou em uma baixa acurácia. Para resolver este problema, decidimos aumentar o tamanho da árvore, para assim aumentar o índice de acerto.

Outra coisa que fizemos foi aplicar um número mínimo de amostras nos nódulos internos e nas folhas, assim melhorando a árvore. Optamos por utilizar o valor mínimo de 1% dos dados, ou 301 amostras. Com isso, a nossa acurácia foi melhorada, além de proporcionar classificações mais confiáveis.

Regressão Logística

O algoritmo de Regressão Logística é um classificador binário e por isso teve um desempenho muito bom na nossa base de dados, sendo um dos modelos com maior acurácia.

Devido ao número de homens ser maior do que o de mulheres, escolhemos modificar os pesos das classes correspondente para equilibrar essa diferença de proporção de dados para que pudéssemos fazer uma logística mais realista, o que aumentou a acurácia do modelo.

KNN

Na aplicação do KNN mudamos a métrica para euclidean e após vários testes chegamos ao melhor resultado com 24 vizinhos, depois aplicamos o algoritmo ball tree, pois através do controle do leaf size conseguimos aumentar ainda mais a acurácia.

Naive Bayes

Para o algoritmo Naive Bayes nós testamos os modelos Gaussian, Multinomial e o Bernoulli. Acreditávamos que o Bernoulli fosse apresentar o melhor resultado por ele ser adequado a bases com recursos binários, porém o modelo Multinomial trouxe um resultado significativamente melhor.

Depois que alteramos a prioridade das classes e deixamos o alpha em 0.1 o que melhorou a acurácia.

Redes Neurais

Com o Algoritmo de Redes Neurais descobrimos que o número de perceptrons da hidden layer em 8 neurônios trouxe o melhor resultado, porém eles eram aleatórios então deixamos o random state em 1 e por fim adicionamos o batch size em 301 (seria 1% da base) o que fez com que a acurácia aumentasse significativamente.

Resultado

Algoritmos	Precision		Recall		F1-Score		Acurácia
	0	1	0	1	0	1	
Árvore de Decisões	**0.74**	0.87	0.74	**0.87**	0.74	0.87	83.00%
Regressão Logística	0.73	**0.93**	**0.87**	0.84	**0.79**	0.88	84.92%
KNN	0.73	0.89	0.78	0.86	0.76	0.87	83.32%
Naive Bayes	**0.74**	0.90	0.80	**0.87**	0.77	0.88	84.23%
Redes Neurais	0.74	0.91	0.84	0.86	**0.79**	**0.89**	**85.14%**

Os algoritmos com melhores resultados foram o de Regressão Logística que trabalhou perfeitamente com os dados binários e as Redes Neurais que apresentou resultados superiores em quase todos os testes.

Os modelos de Árvore de Decisões e KNN foram os que apresentaram os piores resultados de acurácia, mas a Árvore de Decisões ainda se destacou em Recall e Precision enquanto o KNN não trouxe nenhum resultado acima da média.

Redes Neurais tem uma acurácia um pouco maior, porém ela tem alguns resultados inferiores no Recall e Precision, porém o que definiu ela como o algoritmo vencedor foi seu resultado superior no F1-Score.