PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

JAMES GIL BRITO DE SOUSA

MACHINE LEARNING PARA MODELAGEM DO PERFIL DAS FAMÍLIAS DE EXTREMA POBREZA DA CIDADE DE BELO HORIZONTE

Belo Horizonte
Abril de 2022

JAMES GIL BRITO DE SOUSA

MACHINE LEARNING PARA MODELAGEM DO PERFIL DAS FAMÍLIAS DE EXTREMA POBREZA DA CIDADE DE BELO HORIZONTE

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Belo Horizonte

Abril de 2022

SUMÁRIO

1. Introdução

1.1. Contextualização

O Cadastro Único é um conjunto de informações sobre as famílias brasileiras em situação de pobreza e extrema pobreza.

Essas informações são utilizadas pelo Governo Federal, pelos Estados e pelos municípios para implementação de políticas públicas capazes de promover a melhoria da vida dessas famílias. Devem estar cadastradas as famílias de baixa renda que ganham até meio salário mínimo por pessoa ou que ganham até 3 salários mínimos de renda mensal total.

Diversos programas e benefícios sociais do Governo Federal utilizam o Cadastro Único como base para seleção das famílias. Esses programas são:

Programa Auxílio Brasil;
Programa Minha Casa, Minha Vida;
Bolsa Verde – Programa de Apoio à Conservação Ambiental;
Programa de Erradicação do Trabalho Infantil – PETI;
Fomento – Programa de Fomento às Atividades Produtivas Rurais;
Carteira do Idoso;
Aposentadoria para pessoa de baixa renda;
Programa Brasil Carinhoso;
Programa de Cisternas;
Telefone Popular;
Carta Social;
Pro Jovem Adolescente;
Tarifa Social de Energia Elétrica;
Passe Livre para pessoas com deficiência;
Isenção de Taxas em Concursos Públicos.

Os Estados e municípios também utilizam os dados do Cadastro Único como base para seus programas sociais.

O município promove visitas domiciliares às famílias de baixa renda periodicamente para efetuar o cadastramento. Para que a família possa ser cadastrada, é importante ter uma pessoa responsável pela família para responder às perguntas do cadastro. Essa pessoa deve fazer parte da família, morar na mesma casa e ter pelo menos **16 anos.**

Para o responsável pela família, de **preferência uma mulher**, é necessário o CPF ou Título de Eleitor. Exceção: no caso de responsável por famílias indígenas e quilombolas, pode ser apresentado qualquer um dos documentos como Certidão de Nascimento, Certidão de Casamento, CPF, Carteira de Identidade (RG), Certidão Administrativa de Nascimento do Indígena (RANI), Carteira de Trabalho ou Título de Eleitor.

1.2. Entendimento do Problema

Este trabalho visa entender melhor o perfil das pessoas no Cadastro Único de Belo Horizonte, consequentemente, entender o perfil das pessoas de baixa renda.

Este entendimento pode parecer óbvio a princípio, se for analizado somente a renda, que é um dado numérico direto e critério para estar nesse cadastro, mas segundo o viés da sociedade, existem estigmas sociais que podem não corresponder com a realidade do dado descritivo.

1.3. Solução Proposta

Utilizando técnicas de Machine Learning os algoritmos podem aprender de forma a identificar os padrões(features) mais persistentes que levam a pobreza extrema ou são características das pessoas de baixa renda. Sendo assim, pode ser um ponto de partida para uma tomada de decisão, com relação a liberação de benefícios sociais, cálculo de cotas mais justas, ajuda humanitária, mecanismos de combate a pobreza e apoio a saúde de um determinado perfil. Esses benefícios podem ser melhor aplicados tendo em vista um modelo de previsão mais acertivo sem viés humano.

2. Metodologia Aplicada

2.1. Linguagens e Bibliotecas Utilizadas

- ***ETL*:** A técnica de extração e tratamento dos dados recebidos em CSV foram executados via linguagem de programação ***PYTHON*.**
- ***Pandas***: Biblioteca para criação de um Dataframe, e tratamento inicial.
- ***Seaborn:*** Utilizado para geração de gráficos e análise exploratória.
- **Scikit-Learn*:*** Principal biblioteca de Machine Learning.
- ***Cross Validation:*** Técnica utilizada através do **Scikit-Learn** para separação da base de treinamento e teste e validação da mesma.

2.2. Algoritmos de Machine Learning Implementados

Foram utilizados algoritmos supervisionados, usados para problemas de classificação binária (quando o destino é categórico).

- ***LR: Logistc Regression,*** a regressão logística usa essencialmente uma função logística para modelar uma variável de saída binária.
- ***KNN: K-Nearest Neighbors,*** classificador onde o aprendizado é baseado “no quão similar” é um dado (um vetor) do outro. O treinamento é formado por vetores de n dimensões.
- ***CART: Decision Tree Classifier*** é um lgoritmo de Aprendizado de Máquina Supervisionado que usa um conjunto de regras para tomar decisões, de forma semelhante à forma como os humanos tomam decisões.
- ***NB: Gaussian Naive Bayes*** é um modelo de classificação probabilística básico.
- ***RF: Random Forest Classifier*** é um meta estimador que ajusta vários classificadores de árvore de decisão em várias subamostras do conjunto de dados e usa a média para melhorar a precisão preditiva e controlar o ajuste excessivo.

3. Origem dos Dados Utilizados

De acordo com a Open Knowledge Foundation, dados abertos são dados que podem ser livremente acessados, utilizados, modificados e compartilhados por qualquer pessoa, estando sujeitos, no máximo, a exigências que visem preservar sua proveniência e abertura (por exemplo, citação da fonte ou divulgação com a mesma licença).

O Portal Brasileiro de Dados Abertos é a ferramenta disponibilizada pelo governo para que todos possam encontrar e utilizar os dados e as informações públicas. O portal preza pela simplicidade e organização para que você possa encontrar facilmente os dados e informações que precisa. O portal também tem o objetivo de promover a interlocução entre atores da sociedade e com o governo para pensar a melhor utilização dos dados, promovendo impactos positivos sob os pontos de vista social e econômico.

A Lei nº 12.527, sancionada em 18 de novembro de 2011, tem o propósito de regulamentar o direito constitucional de acesso dos cidadãos às informações públicas e seus dispositivos são aplicáveis aos três Poderes da União, Estados, Distrito Federal e Municípios.

A política de dados abertos do Poder Executivo Federal regula e orienta a publicação de dados abertos governamentais pelos órgãos e entidades do Governo Federal. Estão aqui incluídos os órgãos públicos, autarquias e fundações públicas.

O Plano de Dados Abertos (PDA) orienta as ações de implementação e promoção de abertura de dados, inclusive geoespacializados, de uma organização. O documento faz parte da Política de Dados Abertos do Poder Executivo Federal e seu objetivo é organizar e padronizar os processos de publicação de dados abertos do estado, resultando em maior disponibilidade, acesso, qualidade e ampla reutilização dos dados abertos pelas partes interessadas, tanto na sociedade e quanto na própria administração pública.

A Infraestrutura Nacional de Dados Abertos é o conjunto de padrões, tecnologias e orientações para disseminação e compartilhamento de dados e informações públicas em formato aberto. Os padrões e orientações são aprovados pelo Comitê Gestor, que é composto por órgãos e entidades do Poder Executivo Federal, e conta com a participação da sociedade civil e da academia.

A utilização de Aprendizado de Máquina com essa base pode revelar para possíveis análises futuras o perfil das pessoas de baixa renda, quais são as características que mais se agrupam nesse conjunto de pessoas, e pode auxiliar aos planos de governo a tomada de decisão sobre melhorias da sociedade em aspéctos que podem reforçar a melhoria de renda, planos mais efetivos para solucionar a pobreza ou mesmo uma melhor filtragem e verificação de fraude em cadastro.

4. Coleta de Dados

A base de dados utilizada foi adquirida pelo site **Dados Abertos do Governo Federal**, de licença **Creative Commons** (cc-by). Sendo esta base de âmbito municipal cedida pela prefeitura de **Belo Horizonte**. A data mais atualizada desta base foi de setembro de **2021**, em formato csv com **483.804 registros**. Os dados foram adquiridos via arquivo CSV, disponibilizados pelo portal Dados Abertos do Governo, foram estruturados em um Dataframe utilizando o Pandas é uma biblioteca para uso em Python, open-source e de uso gratuito (sob uma licença BSD).

5. Tratamento dos Dados

5.1. Limpeza dos dados

Os dados normalmente precisam ser analizados sob o aspecto de campos nulos, duplicados ou inválidos. Abaixo as informações iniciais.

```
df = pd.DataFrame()
for chunk in pd.read_csv('Cadastro_unico_BH.csv', sep = ';', na_values = '?', chunksize = 500000):
    df = pd.concat([df,chunk])
```

```
df.head(20)
```

	IDADE	GRAU_INSTRUCAO	PARENTESCO_RF	COR_RACA	SEXO	BOLSA_FAMILIA	POP_RUA	FAIXA_RENDA_FAMILIAR_PER_CAPITA
0	52.0	Fundamental incompleto	CONJUGE OU COMPANHEIRO(A)	Parda	MASCULINO	SIM	NAO	Ate R$89.00
1	52.0	Fundamental incompleto	PESSOA RESPONSAVEL PELA UNIDADE FAMILIAR RF	Branca	FEMININO	SIM	NAO	Ate R$89.00
2	19.0	Fundamental incompleto	FILHO (A)	Branca	MASCULINO	SIM	NAO	Ate R$89.00
3	20.0	Fundamental completo	FILHO (A)	Parda	FEMININO	SIM	NAO	Ate R$89.00
4	50.0	Fundamental incompleto	PESSOA RESPONSAVEL PELA UNIDADE FAMILIAR RF	Parda	FEMININO	SIM	NAO	Ate R$89.00
5	18.0	Fundamental completo	FILHO (A)	Parda	MASCULINO	SIM	NAO	Ate R$89.00
6	4.0	Nao Informado	NETO(A) OU BISNETO (A)	Branca	MASCULINO	SIM	NAO	Ate R$89.00

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 483803 entries, 0 to 483802
Data columns (total 8 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   IDADE                           483802 non-null  float64
 1   GRAU_INSTRUCAO                  483803 non-null  object 
 2   PARENTESCO_RF                   482461 non-null  object 
 3   COR_RACA                        483803 non-null  object 
 4   SEXO                            483803 non-null  object 
 5   BOLSA_FAMILIA                   483803 non-null  object 
 6   POP_RUA                         483803 non-null  object 
 7   FAIXA_RENDA_FAMILIAR_PER_CAPITA 483803 non-null  object 
dtypes: float64(1), object(7)
memory usage: 29.5+ MB
```

Figura 1-2 - Informação geral do dataframe.

Foi então necessário a retirada das linhas que continham dados outliers, quer sejam de preenchimento duvidoso ou que de alguma forma foram indicados erroneamente. Um dado crítico seria a idade mínima para o responsável pela família, não sendo aceitável idade inferior à 16 anos. Listados abaixo os unícos dados originalmente numéricos.

```
# Exclui todos registros em que a idade mínima não foi respeitada
indexNames = df[ df['IDADE'] < 16 ].index
df.drop(indexNames , inplace=True)
```

```
# Variáveis numéricas.
df.describe()
```

	IDADE
count	348265.000000
mean	42.631579
std	18.921215
min	16.000000
25%	26.000000
50%	40.000000
75%	57.000000
max	115.000000

Figura 3 - Análise sobre idade.

Após a exclusão dos registros com idade abaixo de 16 anos, teve-se uma redução da base, mas que ainda mantém um tamanho considerável de **348.265** linhas de **483.803** linhas. Uma exclusão de **135.538** registros. Uma análise superficial indica um problema na forma com que esses dados foram cadastrados, ou até mesmo um indicativo de alerta sobre esse cadastro.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 348265 entries, 0 to 483802
Data columns (total 8 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   IDADE                           348265 non-null  float64
 1   GRAU_INSTRUCAO                  348265 non-null  object 
 2   PARENTESCO_RF                   348265 non-null  object 
 3   COR_RACA                        348265 non-null  object 
 4   SEXO                            348265 non-null  object 
 5   BOLSA_FAMILIA                   348265 non-null  object 
 6   POP_RUA                         348265 non-null  object 
 7   FAIXA_RENDA_FAMILIAR_PER_CAPITA 348265 non-null  object 
dtypes: float64(1), object(7)
memory usage: 23.9+ MB
```

Figura 4 - Visualização da informação dos dados após limpeza.

Abaixo listagem dos dados que são categóricos.

```
# Variáveis categóricas.
df.describe(include=['O'])
```

	GRAU_INSTRUCAO	PARENTESCO_RF	COR_RACA	SEXO	BOLSA_FAMILIA	POP_RUA	FAIXA_RENDA_FAMILIAR_PER_CAPITA
count	348265	348265	348265	348265	348265	348265	348265
unique	7	11	6	2	2	2	4
top	Fundamental incompleto	PESSOA RESPONSAVEL PELA UNIDADE FAMILIAR RF	Parda	FEMININO	NAO	NAO	Ate R$89.00
freq	125102	195296	207207	211830	228415	339645	115159

Figura 5 - Visualização dos campos categóricos.

5.2. Pré-processamento dos Dados

Como os algoritmos que serão utilizados, necessitam de dados numéricos, foram feitas as transformações dos mesmos.

```
# Trasnformando as variáveis categóricas em numéricas.
cols = df[df.select_dtypes(['object']).columns]

for c in cols:
    encoding = LabelEncoder()
    encoding.fit(list(df[c].values))
    df[c] = encoding.transform(list(df[c].values))
df = pd.get_dummies(df)
```

```
df.head(20)
```

	GRAU_INSTRUCAO	PARENTESCO_RF	COR_RACA	SEXO	BOLSA_FAMILIA	POP_RUA	FAIXA_RENDA_FAMILIAR_PER_CAPITA
0	1	0	4	1	1	0	1
1	1	9	1	0	1	0	1
2	1	2	1	1	1	0	1
3	0	2	4	0	1	0	1
4	1	9	4	0	1	0	1
5	0	2	4	1	1	0	1
7	1	0	5	1	0	0	2
8	1	9	5	0	0	0	2
9	2	2	1	1	1	0	1
10	0	9	4	0	1	0	1
11	0	0	4	1	1	0	1
12	1	2	4	1	1	0	1
13	1	2	4	1	1	0	1

Figura 6-7 - Trecho da codificação para conversão do novo dataframe numérico.

6. Processos de Machine Learning

6.1. Cross Validation

A maneira mais utilizada e aceita para avaliar modelos treinados via dados é utilizando **cross-validation** (ou validação cruzada).

A validação cruzada é usada principalmente no aprendizado de máquina aplicado para estimar a habilidade de um modelo de aprendizado de máquina em dados não vistos. Ou seja, usar uma amostra limitada para estimar o desempenho do modelo em geral quando usado para fazer previsões sobre dados não usados durante o treinamento do modelo.

Aprender os parâmetros de uma função de previsão e testá-la nos mesmos dados é um erro metodológico: um modelo que apenas repetiria os rótulos das amostras que acabou de ver teria uma pontuação perfeita, mas não conseguiria prever nada de útil ainda. dados não vistos. Essa situação é chamada de overfitting.

Para evitar isso, é uma prática comum ao realizar um experimento de aprendizado de máquina (supervisionado) manter parte dos dados disponíveis como um conjunto de teste. A palavra “experiência” não pretende denotar apenas uso acadêmico, porque mesmo em ambientes comerciais o aprendizado de máquina geralmente começa experimentalmente.

Logo mais um fluxograma do fluxo de trabalho típico de validação cruzada no treinamento do modelo. Os melhores parâmetros podem ser determinados por técnicas de busca em grade X_test, y_test.

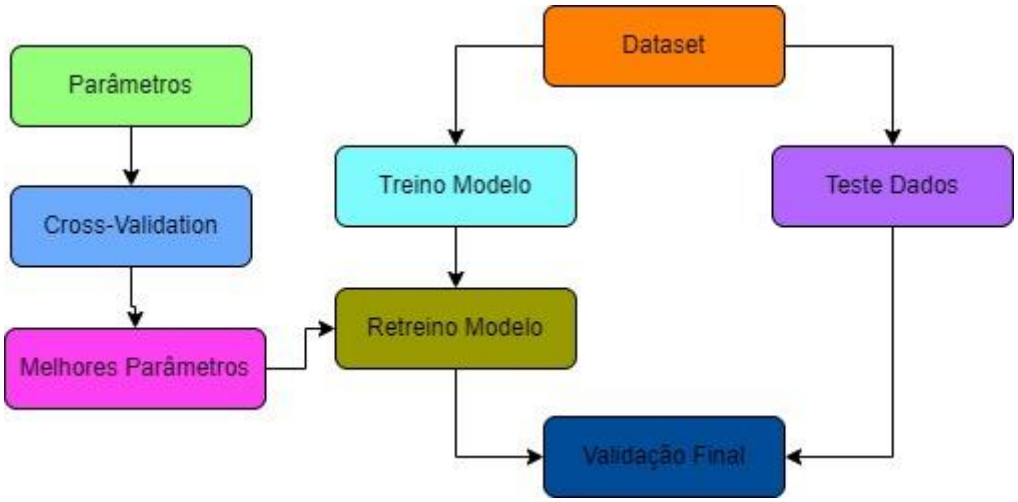

Figura 8 – Fluxograma de Cross-Validation. Autoria Própria.

No scikit-learn, uma divisão aleatória em conjuntos de treinamento e teste pode ser calculada rapidamente com a train_test_split função auxiliar.

Ao avaliar diferentes configurações (“hiperparâmetros”) para estimadores, como a configuração que deve ser definida manualmente para um SVM, ainda existe o risco de overfitting no conjunto de teste porque os parâmetros podem ser ajustados até que o estimador tenha um desempenho ideal. Dessa forma, o conhecimento sobre o conjunto de testes pode “vazar” para o modelo e as métricas de avaliação não relatam mais o desempenho da generalização. Para resolver este problema, outra parte do conjunto de dados pode ser mantida como o chamado “conjunto de validação”: o treinamento prossegue no conjunto de treinamento, após o que a avaliação é feita no conjunto de validação e quando o experimento parece ser bem-sucedido , a avaliação final pode ser feita no conjunto de teste.

Uma solução para este problema é um procedimento chamado validação cruzada (CV para abreviar). Um conjunto de teste ainda deve ser mantido para avaliação final, mas o conjunto de validação não é mais necessário ao fazer o CV. Na abordagem básica, chamada k -fold CV, o conjunto de treinamento é dividido em k conjuntos menores (outras abordagens são descritas abaixo, mas geralmente

seguem os mesmos princípios). O seguinte procedimento é seguido para cada uma das k “dobras”:

Um modelo é treinado usando das dobras como dados de treinamento e o modelo resultante é validado na parte restante dos dados (ou seja, é usado como um conjunto de teste para calcular uma medida de desempenho, como precisão).

A medida de desempenho relatada pela validação cruzada k -fold é então a média dos valores calculados no loop. Essa abordagem pode ser computacionalmente cara, mas não desperdiça muitos dados (como é o caso ao se fixar um conjunto de validação arbitrário), o que é uma grande vantagem em problemas como inferência inversa, onde o número de amostras é muito pequeno.

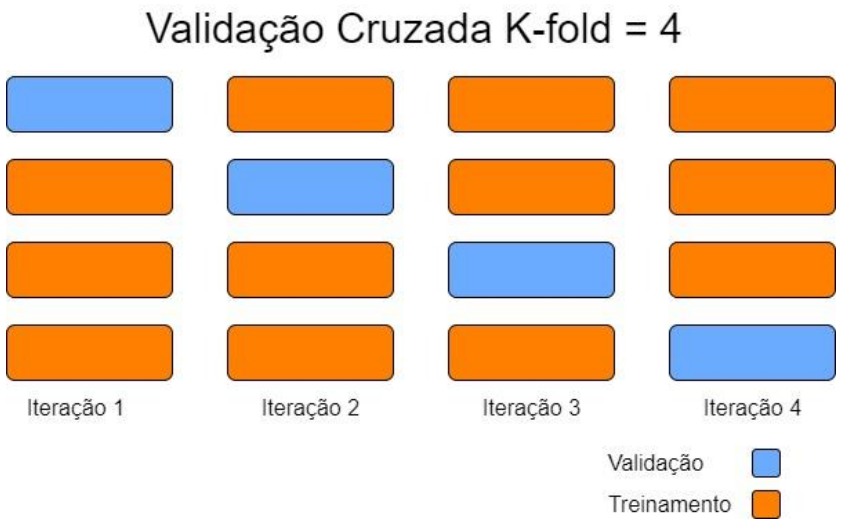

Figura 9 – Fluxograma de K fold. Autoria Própria.

O procedimento tem um único parâmetro chamado k que se refere ao número de grupos em que uma determinada amostra de dados deve ser dividida. Como tal, o procedimento é frequentemente chamado de validação cruzada k-fold. Quando um valor específico para k é escolhido, ele pode ser usado no lugar de k na referência ao modelo, como k=10 tornando-se validação cruzada de 10 vezes.

```
# Criando o modelo
modelo_random = RandomForestClassifier()

# Definindo K
kfold = StratifiedKFold(n_splits=10, shuffle = True, random_state=7)
```

Figura 10 - Trecho da codificação para determinação kfold.

Essa abordagem envolve dividir aleatoriamente o conjunto de observações em k grupos, ou dobras, de tamanho aproximadamente igual. A primeira dobra é tratada como um conjunto de validação e o método é ajustado nas k − 1 dobras restantes.

A escolha de k geralmente é 5 ou 10, mas não há regra formal. À medida que k aumenta, a diferença de tamanho entre o conjunto de treinamento e os subconjuntos de reamostragem diminui. À medida que essa diferença diminui, o viés da técnica se torna menor.

6.2. Scikit-Learn - Biblioteca utilizada em Python

A biblioteca utilizada para o propósito deste trabalho foi o **Scikit-Learn.** O **Scikit-Learn** é uma das bibliotecas de Machine Learning mais conhecidas e utilizadas do Python, dentre as diversas existentes, elaborada em código aberto e desenvolvida para suportar e possibilitar o treino de diversas técnicas de estatística e Machine Learning, para aprendizagem supervisionada e não supervisionada.

O Scikit-Learn fornece ferramentas importantes para os vários momentos do ciclo de projetos de Machine Learning, como:

Datasets: disponibiliza alguns datasets que podem ser baixados para o projeto com poucos comandos, como o dataset Iris, um dos mais conhecidos da área de reconhecimento de padrões.

Pré-processamento de dados: fornece diversas técnicas de preparação de dados, como normalização e encoding.

Modelos: implementa diversos modelos de Machine Learning, tais como Regressão Linear, SVM e Random Forest, possibilitando o ajuste, avaliação e seleção do melhor modelo para o problema.

Utilizando o recurso de pipelines, foi criado uma linha base de desempenho para verificação de vários modelos diferentes com suas configurações padrão dentro do pacote Scikit-Learn. Os modelos utilizados estão no próximo código:

```
pipelines = []
pipelines.append(('Scaled-LR', Pipeline([('Scaler', StandardScaler()),('LR', LogisticRegression())])))
pipelines.append(('Scaled-KNN', Pipeline([('Scaler', StandardScaler()),('KNN', KNeighborsClassifier())])))
pipelines.append(('Scaled-CART', Pipeline([('Scaler', StandardScaler()),('CART', DecisionTreeClassifier())]))),
pipelines.append(('Scaled-NB', Pipeline([('Scaler', StandardScaler()),('NB', GaussianNB())]))),
#pipelines.append(('Scaled-SVM', Pipeline([('Scaler', StandardScaler()),('SVM', SVC())])))
pipelines.append(('Scaled-RF', Pipeline([('Scaler', StandardScaler()),('RF', RandomForestClassifier())])))
resultados = []
nomes = []
```

Figura 11 - Trecho da codificação para teste de cada algoritmo.

Obtendo os resultados seguintes:

```
Scaled-LR: 0.626546 (0.002541)
Scaled-KNN: 0.577658 (0.024157)
Scaled-CART: 0.632053 (0.002940)
Scaled-NB: 0.568848 (0.001471)
Scaled-RF: 0.632033 (0.002955)
```

Figura 12 - Trecho da codificação com resultado preliminar de cada algoritmo.

Estes resultados sugerem que os algoritmos ***Decision Tree Classifier e Random Forest Classifier*** têm potencial de serem bons modelos para o dataset proposto.

Estes são apenas valores médios de acurácia, então foi observado a distribuição dos resultados de cada fold da validação cruzada, comparando os modelos usando boxplots. Esse é um tipo de gráfico com muita informação sobre os dados: Nele pode-se ver o limite inferior e superior, os quartis, a mediana, como os dados estão distribuídos e os possíveis outliers.

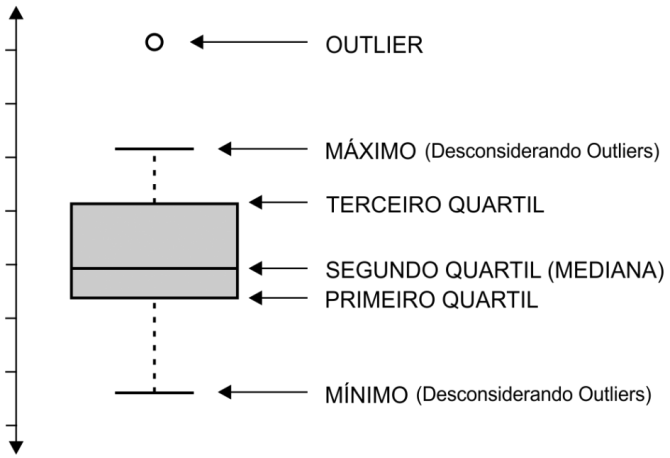

Figura 13 – Imagem para entendimento do boxplot.

Quartis: Os quartis dividem os dados em quatro partes. Ordenando os dados cada quartil vai possuir 25% das observações. Ou seja, os primeiros 25% das observação estarão até o corte do 1º quartil, os próximos 25% até o corte do 2º quartil, etc.

Limites: O limite inferior é calculado sendo, geralmente, 1,5 vezes o corte do 1º quartil e o limite superior sendo 1,5 vezes o corte do 3º quartil.

O corte do segundo quartil é igual a mediana.

Outliers são observações atípicas, muito afastadas da maioria dos dados. Num boxplot esses valores ficam abaixo do limite inferior ou acima do limite superior.

```
fig = plt.figure()
fig.suptitle('Comparando os Algoritmos Padronizados')
ax = fig.add_subplot(111)
plt.boxplot(resultados)
ax.set_xticklabels(nomes)
plt.show()
```

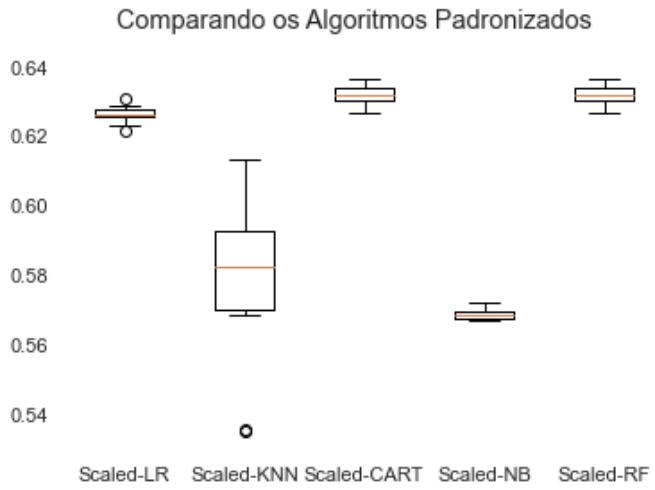

Figura 14 - Trecho da codificação para confirmação do desempenho de cada algoritmo.

6.3. Parâmetros de Ajuste

Foram reajustados os algoritmos ***Decision Tree Classifier e Random Forest Classifier*** parapossível melhoria de desempenho, já que os mesmos melhor se adequaram a solução proposta.

O critério utilizado para realizar a parametrização é o da utilidade do atributo para a classificação. Aplica-se, por este critério, um determinado ganho de informação a cada atributo. O atributo escolhido como atributo teste para o corrente nó é aquele que possui o maior ganho de informação.

A partir desta aplicação, inicia-se um novo processo de partição. Nos casos em que a árvore é usada para classificação, os critérios de partição mais conhecidos são baseados na entropia e índice Gini (Onoda, 2001).

A **entropia** ajuda a construir uma árvore de decisão apropriada para selecionar o melhor divisor. A entropia pode ser definida como uma medida da pureza da subdivisão. A entropia sempre está entre 0 e 1. A entropia de qualquer divisão pode ser calculada por esta fórmula:

$$H(s) = -P_(+ \log_2 P_(+) - P_(-) \log_2 P_(-)$$

$$\text{Here } P_(+) / P_(-) = \%\text{ of + ve class } 1\%\text{ of - ve class}$$

O funcionamento interno da impureza **Gini** também é um tanto semelhante ao funcionamento da entropia na Árvore de Decisão. No algoritmo da árvore de decisão, ambos são usados para construir a árvore, dividindo de acordo com os recursos apropriados, mas há uma grande diferença no cálculo de ambos os métodos. A impureza de Gini de recursos após a divisão pode ser calculada usando esta fórmula:

$$\text{G I=1-}\sum_{i=1}^{n}(p)^2 \quad GI = 1 - \left[\left(P_(+)\right)^2 + \left(P_(-)\right)^2\right]$$

O funcionamento interno de ambos os métodos é muito semelhante e ambos são usados para calcular o recurso/divisão após cada nova divisão. Mas se compararmos os dois métodos, a Impureza de Gini é mais eficiente do que a entropia em termos de poder de computação.

6.4. Ajuste do Modelo Random Forest

Foram então utilizados os parâmetros para os dois algoritmos e verificado a melhora ou não da acurácia dos mesmos.

```
# Possíveis valores para o critério de divisão
val_criterion = ['gini', 'entropy']
```

Figura 15 - Trecho da codificação para seleção dos critérios de ajuste.

```
Grid scores on development set:
mean:0.63187,std:0.003,params:{'n_estimators': 20, 'criterion': 'gini'}
mean:0.632,std:0.00294,params:{'n_estimators': 50, 'criterion': 'gini'}
mean:0.63201,std:0.00298,params:{'n_estimators': 100, 'criterion': 'gini'}
mean:0.63202,std:0.00297,params:{'n_estimators': 150, 'criterion': 'gini'}
mean:0.63205,std:0.00295,params:{'n_estimators': 200, 'criterion': 'gini'}
mean:0.63201,std:0.00292,params:{'n_estimators': 20, 'criterion': 'entropy'}
mean:0.63205,std:0.00298,params:{'n_estimators': 50, 'criterion': 'entropy'}
mean:0.63203,std:0.00291,params:{'n_estimators': 100, 'criterion': 'entropy'}
mean:0.63201,std:0.00292,params:{'n_estimators': 150, 'criterion': 'entropy'}
mean:0.63204,std:0.00296,params:{'n_estimators': 200, 'criterion': 'entropy'}

Melhor parâmetro:{'n_estimators': 200, 'criterion': 'gini'}, Score:0.632053175635428
```

Figura 16 - Trecho da codificação com melhor resultado de parâmetros do Random Forest.

Os resultados a seguir indicam a melhor parametrização do Random Forest para o trabalho proposto, a utilização do Gini.

6.5. Ajuste do Modelo Decision Tree

As árvores de decisão são ferramentas poderosas de classificação consideradas como uma das técnicas mais eficientes aplicadas em vários campos científicos, tais como Machine Learning e Inteligência Artificial segundo Elouedi (2002). O resultado sugere entropy como melhor parâmetro.

```
Grid scores on development set:
mean:0.63205,std:0.00294,params:{'max_depth': 14, 'criterion': 'gini'}
mean:0.62758,std:0.00227,params:{'max_depth': 3, 'criterion': 'gini'}
mean:0.63205,std:0.00294,params:{'max_depth': 25, 'criterion': 'gini'}
mean:0.63201,std:0.00269,params:{'max_depth': 5, 'criterion': 'gini'}
mean:0.63206,std:0.00294,params:{'max_depth': 24, 'criterion': 'entropy'}
mean:0.63205,std:0.00294,params:{'max_depth': 29, 'criterion': 'gini'}
mean:0.63206,std:0.00294,params:{'max_depth': 17, 'criterion': 'entropy'}
mean:0.63205,std:0.00294,params:{'max_depth': 27, 'criterion': 'gini'}
mean:0.63206,std:0.00294,params:{'max_depth': 16, 'criterion': 'entropy'}
mean:0.57548,std:0.0016,params:{'max_depth': 1, 'criterion': 'gini'}

Melhor parâmetro:{'max_depth': 24, 'criterion': 'entropy'}, Score:0.6320560472182303
```

Figura 17 - Trecho da codificação com melhor resultado de parâmetros do Decision Tree.

7. Análise Exploratória dos Dados

Alguns cruzamentos de dados foram utilizados para verificar a influência de cada feature sobre o modelo proposto. No primeiro gráfico, uma amostragem de quantidade de famílias em certa faixa de renda. As rendas foram coletadas em quatro divisões: Até R$89,00, entre R$89,00 e R$178,00, de R$178,01 até 0,5 sálarios mínimos e acima de 0,5 salário mínimo per-capita.

```
# Distribuição das classes (variável FAIXA_RENDA_FAMILIAR_PER_CAPITA)
sns.catplot(x='FAIXA_RENDA_FAMILIAR_PER_CAPITA',kind = 'count', data = df, orient = "h", height = 5, aspect = 2.2)
```

```
<seaborn.axisgrid.FacetGrid at 0x1cb58137df0>
```

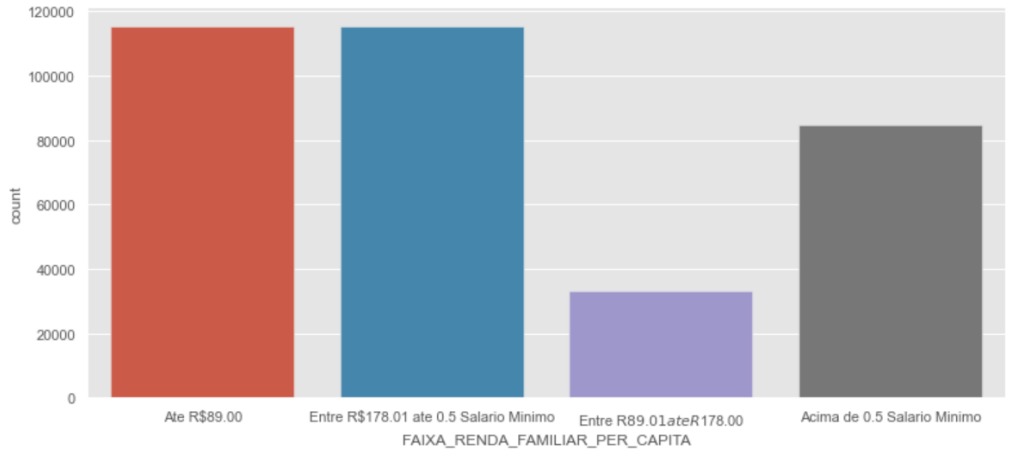

Figura 18 - Trecho da codificação e gráfico de Renda per Capita.

No próximo gráfico está a faixa de renda por escolaridade.

```
# Distribuição das classes por nível de escolaridade
plt.figure(figsize=(20,10))
sns.countplot(data=df, x = 'GRAU_INSTRUCAO',  hue = 'FAIXA_RENDA_FAMILIAR_PER_CAPITA')
plt.xticks(rotation=65)
```

```
(array([0, 1, 2, 3, 4, 5, 6]),
 [Text(0, 0, 'Fundamental incompleto'),
  Text(1, 0, 'Fundamental completo'),
  Text(2, 0, 'Medio completo'),
  Text(3, 0, 'Sem instrucao'),
  Text(4, 0, 'Medio incompleto'),
  Text(5, 0, 'Superior incompleto ou mais'),
  Text(6, 0, 'Nao Informado')])
```

Figura 19 - Trecho da codificação Grau de Instrução x Renda.

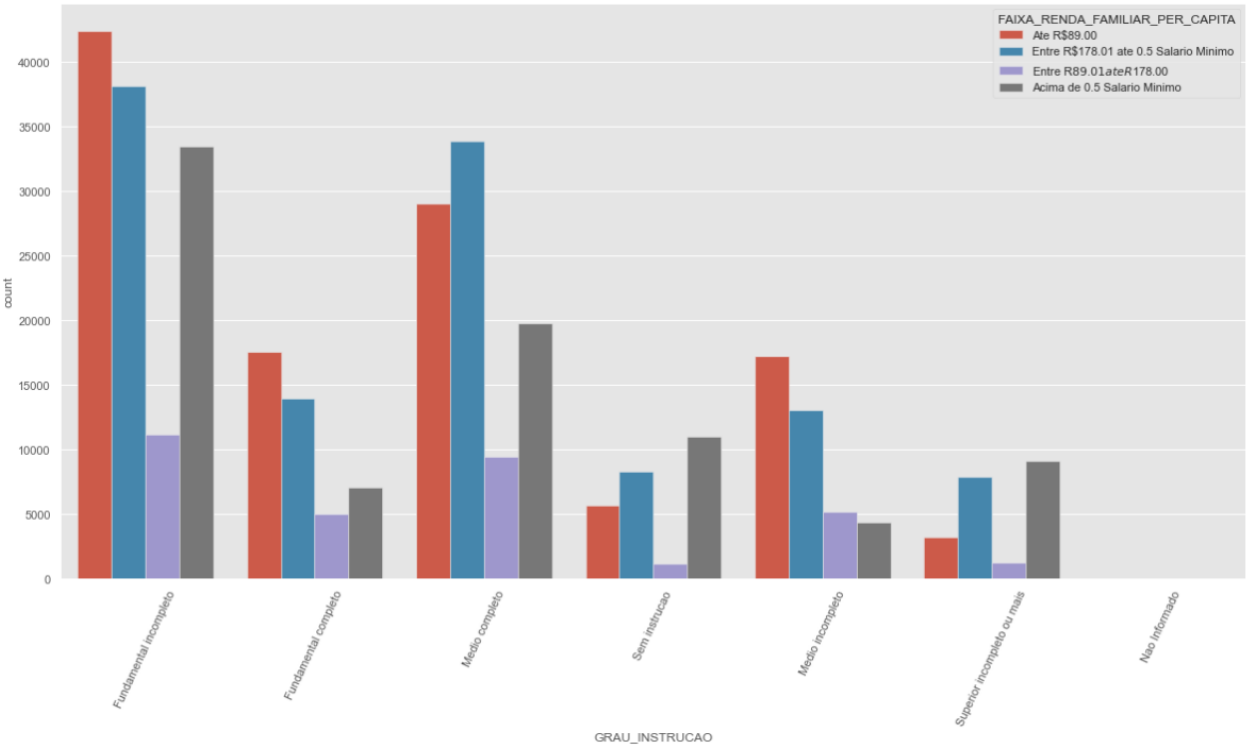

Figura 20 - Trecho do gráfico de Grau de Instrução x Renda.

Também foi analisada a faixa de renda por sexo e por raça, mostrando um ponto de vista um pouco mais abrangente.

```
plt.figure(figsize=(8,6))
sns.countplot(y="SEXO", hue='FAIXA_RENDA_FAMILIAR_PER_CAPITA', data=df)
```

```
<AxesSubplot:xlabel='count', ylabel='SEXO'>
```

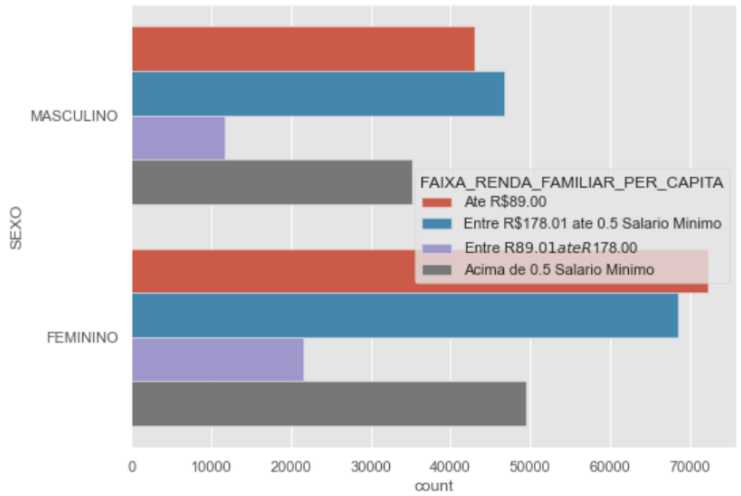

Figura 21 - Trecho da codificação e gráfico Sexo x Renda per Capita.

```
plt.figure(figsize=(8,6))
sns.countplot(y="COR_RACA", hue='FAIXA_RENDA_FAMILIAR_PER_CAPITA', data=df)
```

```
<AxesSubplot:xlabel='count', ylabel='COR_RACA'>
```

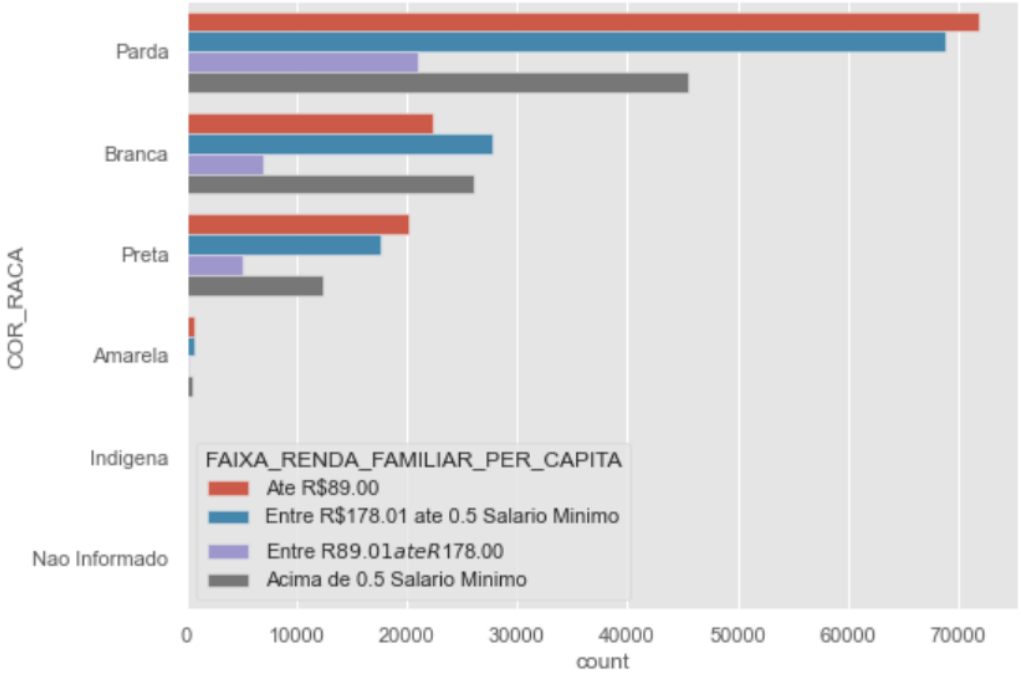

Figura 22 - Trecho da codificação e gráfico Raça x Renda per Capita.

Distribuição de renda por classe de idade:

```
df['IDADE'] = pd.cut(df['IDADE'],[0,25,50,100], labels =['JOVEM','ADULTO','IDOSO'])

# Verificando distribuição.
plt.figure(figsize=(8,6))
sns.countplot(data=df, x = 'IDADE',  hue = 'FAIXA_RENDA_FAMILIAR_PER_CAPITA')
plt.xticks(rotation=70)
```

```
(array([0, 1, 2]),
 [Text(0, 0, 'JOVEM'), Text(1, 0, 'ADULTO'), Text(2, 0, 'IDOSO')])
```

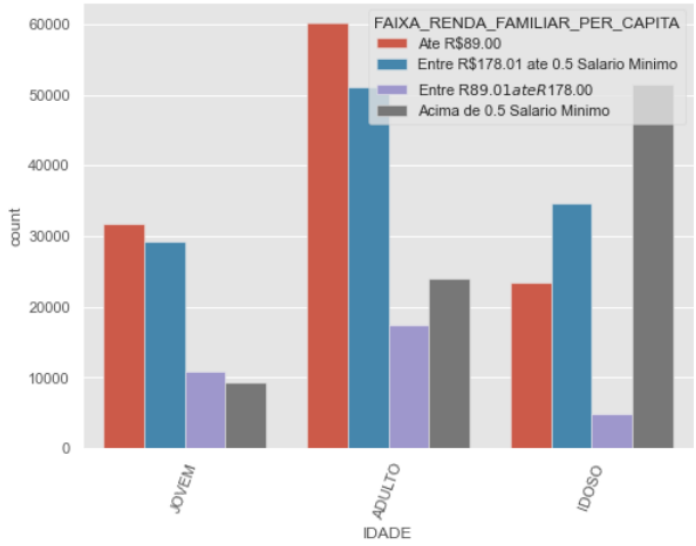

Figura 23 - Trecho da codificação e gráfico Idade x Renda per Capita.

8. Resultados

Para análise de correlação das features(características), é utilizado a matriz de correlação normalizada através de um heatmap. Utiliza-se por padrão a correlação de Pearson, mas não é a mais indicada quando estamos tratando de dados categóricos como a maioria desse dataset, então foi utilizado o heatmap usando a correlação V de Crámer, cujo os coeficiente vão de 0 a 1, só para termos números mais reais.

```
# Análise simultânea da associação entre variáveis.
corr = df.corr()
sns.set(rc={'axes.facecolor':'white', 'figure.facecolor':'white'})
plt.subplots(figsize=(12, 9))
sns.heatmap(corr, vmax=.8,annot_kws={'size': 10}, annot=True, fmt='.2f')
```

```
<AxesSubplot:>
```

Figura 24 - Trecho da codificação e Heatmap da correlação.

Verificado as correlações, pode-se obter um indicativo das features mais responsáveis pelo perfil de critério com relação ao cadastro único. Essas features ou características podem não corresponder como já colocado, ao padrão do que a sociedade espera como características de pessoas de baixa renda.

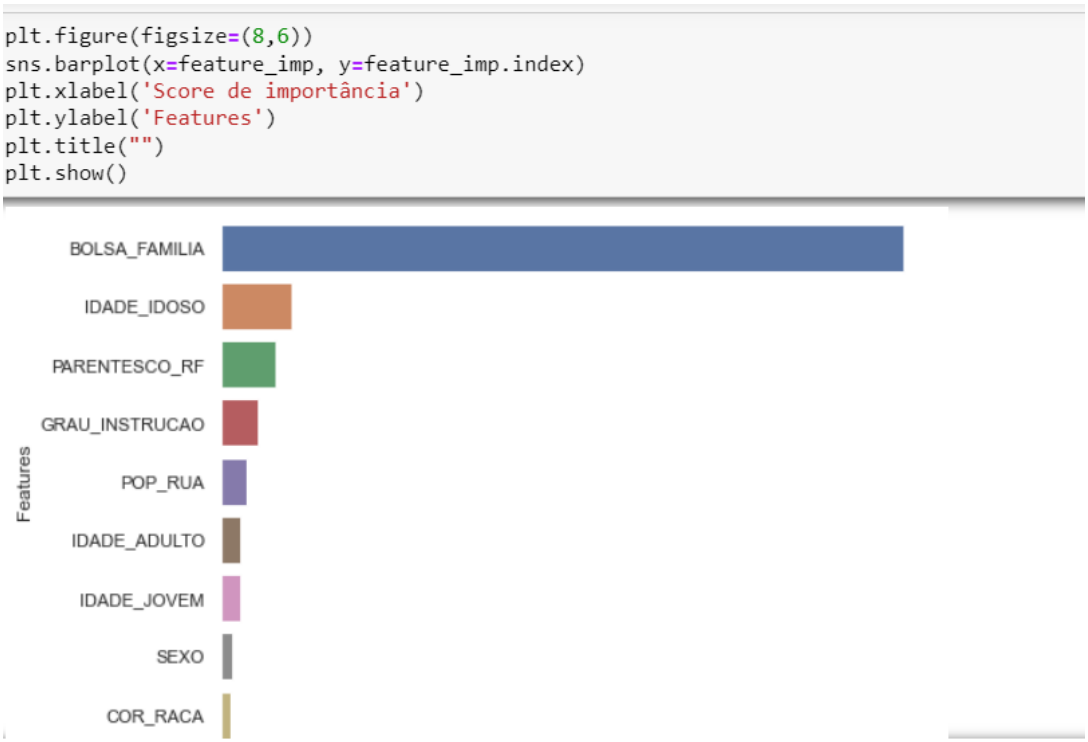

Figura 25 - Features mais importantes do modelo.

O modelo gerado pelo algoritmo foi exportado e pode ser aplicado via WEB para simulação binária em termos percentuais, de uma pessoa ter perfil de baixa renda pelas caraterrísticas apresentadas, ou mesmo o trabalho proposto pode ser uma referência para um modelo mais robusto, onde essas características podem ser um padrão para aquisição de benefício social ou cotas, ou até mesmo verificação da veracidade e robustez do cadastro.

9. Conclusão

O método demonstrado de Machine Learning, através da biblioteca do Scikit-Learn, e do dataset do Cadastro Único, se mostra hábil em reconhecer padrões e características em grupos categóricos, sendo uma metodologia útil para o tipo de problema, tendo em vista uma aplicação para entender o perfil das pessoas de baixa renda, e os pontos sociais que podem ser aprimorados para minimizar a diferença social existente atualmente. Como conclusão imediata, o estudo sugere, que pessoas do sexo feminino de cor parda e adulto são os perfis de maior abrangência. Provavelmente porque a maioria das pessoas responsáveis por lares nesse perfil são mulheres.

Outro fator interessante é o da escolaridade, o estudo sugere e demonstra que a extrema pobreza tende a decair com o aumento do grau de instrução, que sempre foi ponto chave e conhecido da sociedade, mas que se prova de uma maneira mais rosbusta, realmente ser um dos fatores determinantes para futura probreza. Sendo assim, conclui-se que trabalhos nessa direção, tendem a ter aplicações diretas em tomada de decisão, sejam elas do ponto de vista governamental ou do ponto de vista social. Concluindo, é possível aplicações em situações onde o algoritmo pode prever um perfil ou tendência, em diversas áreas, como em áreas jurídicas, para perfil criminal ou de apoio, áreas sociais para recebimento de algum benefício social ou cota, em seguradoras e outras. Minimizando logicamente os possíveis vieses(tendências natuais) ou vícios de cadastro, dataset, ou mesmo da coleta de dados.

10. Links

https://github.com/jamesgilbs/tcc-puc-ai-machine-larning

11. Referências

ELOUEDI, Khaled Mellouli Zied, and Philippe Smets. **A pre-pruning method in belief decision trees.** In The Ninth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU, volume 1, pages 579–586, 2002

ONODA, Mauricio. **Estudo sobre um algoritmo de árvores de decisão acoplado a um sistema de banco de dados relacional.** Dissertação (Mestrado)-Universidade Federal do Rio de Janeiro, Rio de Janeiro 2001.