# Machine Learning Engineer Nanodegree

## Capstone Report

Rodrigo Moreira Fagundes December 23rd, 2018

## I. Definition

### Project Overview

In such a large developing country as Brazil, with so many different economic and cultural setups, tackling modern slavery efficiently is a huge challenge. According the Observatório Digital do Trabalho Escravo (https://observatorioescravo.mpt.mp.br/), from 2003 to 2018, 44,229 people were rescued from degrading working conditions in 3,318 inspections (13.33 rescues per inspaction). 2,006 successful diligences took place in 766 of the 5,570 brazilian municipalities (a coverage of 13.75%). If we add the 1,847 inspections with no rescues, the coverage raises to a 37.9%, with 2,112 locations, but it reveals a success rate of 52.06%. Since 2015, the number of diligences have benn falling, returning to the yearly inspections frequency seen in 2003-2007.

Some regions are not accessible, taking too long to mobilize an inspection - allowing perpetrators to move their operations or just hide their illegal aspects during the audit. Traditionally, the inspections are mobilized based on a denounce and evidences that support it. The most vulnerable people, though, lack the opportunity to reach government agencies.

In opposition, urban centers with high population density have a large number of enterprises to be verified, rendering a traditional coverage goal unrealistic - both due to cost and manpower. Modern slavery practices in urban centers tend to be disguised as just poor labor standards or practices, facing long disputes before estabilishing the perpetration.

### Problem Statement

Given the scenarios, government agencies have to craft a way to concentrate its resources in targets that reach more vulnerable people. One way to create such a prioritization can be trying to identify, using classification, the locations that would most likely to result in more rescues per diligence. Some inputs are very promising for that task, such as municipalities profiles - based on census data provided by IBGE (Instituto Nacional de Geografia e Estatística) - and the record of previous inspections - from MTE (Ministério do Trabalho e Emprego). The output of the classification will indicate municipalities in which inspections may result in maximum rescue rates.

One solution to the problem can be resource optimization by defining high priorities munuicipalities based on statistical inference. By using municipalities similarities and previous diligences data, it's reasonable to focus on locations that are most likely to result in a more effective action, rescuing more people in a single inspection, for instance. By prioritizing municipalities according to the distribution of rescues per inspection, the model can be repeated, hopefully with decreasing numbers of perpetrations.

### Metrics

To measure the success of the solution, the municipalities that have no previous records flagged by the solution should be split in control and test subjects. The testing group should be subject to a task force and the result should be compared to the control (disclosed at the end of the evaluation, to avoid bias). A number of false positives should be expected, and compared to the metrics revealed in the modeling phase.

If the rescues per inspections in the testing group is higher than the control group (error margin considered), the model is proven effective. The result should also be compared to the overall rating and to the previous records - it can reveal a migration of modern slavery practices.

For the purpose of the current study, we'll use the inspections dataset to train, test and validate the model, comparing the results using precision and recall metrics.

## II. Analysis

### Data Exploration

For this study, we're using three datasets. The first is a collection of information on municipality's census, collected by IBGE (Instituto Nacional de Geografia e Estatística), available to the public. The comprehensive dataset, with no relevant missing data so far, is a suitable source for identifying profiles and similarities between locations. Is is also a reliable dataset, once the census conducted by IBGE goes through a rigorous methodology.

The second and third datasets will be the disidentificated registers of operations and inspections. They contain information on the municipalities where inspections took place, how many people were rescued from degrading work conditions, their origin and where they claimed to reside at the moment. In the current study, they'll be used as a base for risk rating, which in turn will become the label for classification.

#### Census

The census data, collected by IBGE (https://www.ibge.gov.br/) was provided by Smartlab (http://smartlab.mpt.mp.br) in December 22nd, 2018 as a CSV file. The dataset is the same used in the Observatório Digital do Trabalho Escravo (http://observatorioescravo.mpt.mp.br). It contains 73 indicators, including GDP, employed population by age, among others. We took the more comprehensive data, from 2010.

In order to remove municipalities' identification, we appended the average of rescues per inspections from the rescues data to the census dataset. The input was generated by MTE (http://mte.gov.br) and provided by Smartlab. It is also used in the Observatório Digital do Trabalho Escravo.

```
import pandas as pd

# Reading the CSV
df = pd.read_csv('data/dataset.csv')

# Formating data from vl_indicador
import re
```

```
df['vl_no_format'] = df['vl_indicador'].map(lambda x: float(re.sub('[\.]', '', str(x))))

# Pivoting the data
df = df.pivot_table(index='cd_mun_ibge', columns='ds_indicador_curto',
values='vl_no_format').reset_index().drop(columns=['cd_mun_ibge'])

# Exploring the data
print(df.shape)
df.head()
```

```
(5565, 113)
```

| ds_indicador_curto | Alfabetização das pessoas de 15 anos ou mais | Analfabetismo das pessoas de 15 a 24 anos | Aprendizes em relação à população ocupada | Area Territorial | Bolsa Família, PETI ou outros programas sociais - Domicílios que recebem | Crescimento da população de 2000 a 2010 | Crianças Ocupadas | Crianças e Adolescentes ocupados como trabalhadores domésticos, sobre o total da população de 10 a 17 anos | Crianças e Adolescentes ocupados no trabalho doméstico | Crianças e adolescentes ocupados | ... | Trabalhadores por conta própria contribuintes de 16 a 64 anos | Trabalho doméstico no total de ocupados de 10 a 17 anos | Valor Adicionado Bruto | Valor Adicionad Bruto a preços correntes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 420.0 | NaN | ... | NaN | NaN | NaN | NaN |
| 1 | 6.544000e+15 | NaN | NaN | NaN | 142882.0 | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |
| 2 | 4.573000e+03 | NaN | NaN | NaN | 5066.0 | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | 7028754.0 | NaN |
| 3 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | 92.0 | 1096192.0 | NaN |
| 4 | NaN | NaN | NaN | NaN | 6049.0 | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN |

5 rows × 113 columns

The census data provided don't seem to be as thorough as expected. It should be revisited in the future studies, should the density proves too low to provide a strong profile for the municipality.

```
# Checking sparsity/density
sdf = df.to_sparse()
print(sdf.density)
```

```
0.1546629137545818
```

Since the dataset revealed itself too sparse, we'll conduct the study using just the rescues dataset. It contains a high dimensionality (higher than the census), because of the depth and granularity of some dimensions.

#### Rescues data

The inspection data was colected from the Observatório Digital do Trabalho Escravo (http://observatorioescravo.mpt.mp.br) in December 22nd, 2018 as a CSV file. It contains register from rescues with high dimensionality, since the indicators were built taking into account the munber of responses of each rescuee survey and includes info on gender, race, instruction, age, occupation (current and desired), among others.

All data are numeric indicatiors, with the ammount of people. They can be divided into 3 main segments:

- '_rgt_' marks the data from rescues or rescuees;
- '_res_' reveals information about the people that reside in that place and were rescued anywhere
- '_nat_' means the amount of people born in a municipality, regardless of where they were rescued

Another important variable is *ds_agregacao_primaria*, which is a second level of granularity, an specialization of the indicator.

**For example:** *te_res_raca* with ds_agregacao_primaria 'Branca' in the municipality 0000000 is an indicator that counts the number of white people that resided in the city 0000000, regardless of where they were rescued.

```
# Reading the CSV
df = pd.read_csv('data/dataset_resgates.csv')

# Setting a default level to summarized indicators
df['ds_agreg_primaria'] = df['ds_agreg_primaria'].fillna('default')
```

```
# Pivoting the data
# Since the data come from surveys, the absence of information means no occurence of that instance, thus the zero-
filling.
df = df.pivot_table(
    index='cd_mun_ibge',
    columns=['cd_indicador','ds_agreg_primaria'],
    values='vl_indicador',
    fill_value=0
).reset_index().drop(columns=['cd_mun_ibge'])

# Exploring the data
print(df.shape)

sdf = df.to_sparse()
print(sdf.density)

df.head()
```

```
(3371, 1518)
1.0
```

| cd_indicador | te_insp_rgt | te_inspecoes | te_nat | te_nat_cnae | | | | | | | ... | te_res_raca | te_res_raca_idade | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ds_agreg_primaria | default | default | default | Abate de suínos, aves e outros pequenos animais | Administração pública em geral | Aluguel de máquinas e equipamentos não especificados anteriormente | Aparelhamento e outros trabalhos em pedras | Apicultura | Armazenamento | Atividade Medica Ambulatorial com Recursos para Realizacao de Procedimentos Cirurgicos | ... | Pessoa Que Se Enquadrar Como de Raça Amarela ( de Origem Japonesa, Chinesa, Coreana, Etc) | >Não Informado | Pessoa Que Se Enquadrar Como Branca | Pessoa Que Se Enquadrar Como Parda ou Se Declarar Como Mulata, Cabocla, Cafuza, Mameluca ou Mestiça de Preto com Pessoa de Outra Cor ou Raça |
| **0** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |

5 rows × 1518 columns

### Algorithms and Techniques

The census data holds a high dimensionality. In order to keep the explainability of the resulting model, we'll use random forests to understand how the dimensions in census data contribute to the model. Beforehand, all municipalities will be disidentificated for the purpose of this study.

We'll estabilish three menace rating (LOW, MEDIUM, HIGH), using the register of previous inspections using the distribution of rescues per inspection. The levels will be, LOW, MEDIUM and HIGH, from the terciles of the selected indicator.

Since there's no previous definition of such rating and we're establishing the levels according to a normal distribution, the labeling class should be balanced (each tercile holds a third of the data). After that, we'll label the municipalities that actually had inspections according to these ratings and the average of the rescues per inspection in them. This rating will be added to the census data as the label for the classification later on.

With the dimensionality reduced and labels set, we'll revisit the sliced dataset, remove the extreme outliers (above Q1 - 3*IQ, where Q1 is the first quartile and IQ is the interquartile) and normalize the data. Then we'll apply Random Forests for classification.

The classification algorithm was chosen because, as an ensemble method, it combines machine learning techniques, reducing biases, improving generalization and decreasing variance. Additionally, random forests are robust to outliers. For the current problem, with a small dataset and features already reduced, its weaknesses, such as high complexity and high time and resource consumption, have low impact.

Random Forests consist of training a series of Decision Trees, each using a subset of the data (bagging method) and using them to predict.

A Decision Tree is a method that consists in a succession of splitting in a dataset according to the information that best defines subsets (best predictor), leading to a label/classification (leaf node). For instance, in a dataset of people, to define if a person is a parent, one information with high information gain is "age" - if, say, under 12, that row falls to a subset that leads to NO KID, otherwise it belongs to another branch. After the first splitting is resolved, each is split again according to the second feature whit higher information gain. The tree can be split until no splitting is possible - there's no doubt about the outcome/classification -, a minimum level of certainty is reached or the maximum number of sequential decisions (pre-defined) is reached.

In the Random Forests technique, after the individual predictions made by its Decision Tree take place, the method returns the class that is the mode of the classes outputs of those individual trees. In other words, each Decision Tree "votes" for a label, according to the rules they found, and then the votes are computed by the Forests, resulting in a label decided by the majority (using mode). By doing that, it avoids the overfitting tendency in individual Decision Tree.

### Benchmark

There's no benchmark model available but the actual rating of rescues per inspection exclusively using the traditional resource placing strategies. Therefore, we'll split the data in two equal parts: a subset for training and validating, that will be used as the benchmark model, and another to test the model.

## III. Methodology

### Data Preprocessing

First of all, municipalities with no rescue will be removed from the dataset. As proposed, only those with actual rescue will be subject of the current study.

```
# Removing rows with te_rgt = 0
df = df[df.te_rgt.default != 0]

print(df.shape)

sdf = df.to_sparse()
print(sdf.density)

df.head()
```

```
(753, 1518)
1.0
```

| cd_indicador | te_insp_rgt | te_inspecoes | te_nat | te_nat_cnae | | | | | | | ... | te_res_raca | te_res_raca_idade | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ds_agreg_primaria | default | default | default | Abate de suínos, aves e outros pequenos animais | Administração pública em geral | Aluguel de máquinas e equipamentos não especificados anteriormente | Aparelhamento e outros trabalhos em pedras | Apicultura | Armazenamento | Atividade Medica Ambulatorial com Recursos para Realizacao de Procedimentos Cirurgicos | ... | Pessoa Que Se Enquadrar Como de Raça Amarela ( de Origem Japonesa, Chinesa, Coreana, Etc) | >Não Informado | Pessoa Que Se Enquadrar Como Branca | Pessoa Que Se Enquadrar Como Parda ou Se Declarar Como Mulata, Cabocla, Cafuza, Mameluca ou Mestiça de Preto com Pessoa de Outra Cor ou Raça |
| **6** | 10 | 16 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 2 | 0 | 0 | 0 |
| **11** | 3 | 3 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| **20** | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |
| **22** | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 5 | 0 | 0 | 0 |
| **24** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 |

5 rows × 1518 columns

The dataset has a high dimensionality. In order to perform the model, we need to rationalize it. Before the transformation to CSV, we categorized the age of recuees from numeric bins to 'minor' and 'adult' - this is important to differentiate child labor and slavery of adults.

We'll also remove occupational data. Its granularity is too high, leading to a right-tailed distribution, concentrated near the origin.

```
idx = pd.IndexSlice

occupation = df.loc[idx[:], idx['te_nat_cnae', :]]
occupation_sample = occupation.head(10)
occupation_sample = occupation_sample.loc[:, (occupation_sample != 0).any(axis=0)]

pd.plotting.scatter_matrix(occupation_sample, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
occupation_sample.head(10)
```

| cd_indicador | te_nat_cnae | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ds_agreg_primaria | Atividades de Apoio a Producao Florestal | Comércio varejista de produtos de padaria, laticínio, doces, balas e semelhantes | Construcao de Rodovias e Ferrovias | Criacao de Bovinos para Corte | Cultivo de Arroz | Desdobramento de madeira | Fabricacao de Alcool | Fabricacao de Laticinios | Ignorado | Producao de Ferro-Gusa | Produção florestal - florestas nativas | Servico de Inseminacao Artificial em Animais |
| **6** | 0 | 0 | 0 | 13 | 4 | 0 | 0 | 0 | 3 | 1 | 0 | 6 |
| **11** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 5 | 0 | 0 | 0 |
| **20** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **22** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **24** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **26** | 0 | 0 | 0 | 18 | 2 | 0 | 0 | 1 | 3 | 0 | 0 | 0 |
| **34** | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 4 | 0 | 0 | 0 |
| **36** | 0 | 0 | 0 | 1 | 1 | 5 | 0 | 0 | 1 | 0 | 0 | 0 |
| **41** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **44** | 0 | 0 | 0 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Occupations data is relevant in the context of tackling modern slavery. These data can be confronted with the joined information from CAGED (Cadastro Geral de Empregados e Desempregados) and census data, so that people can be receive instruction for jobs that are actually in demand or can be moved to loactions where there are job openings for the rescuee's desired occupation.

To the current study, though, they don't contributeto the outcome (high granularity and near-zero right-tailed distribution). Therefore, all occupations data will be removed.

```
occup_cols = [col for col in df.axes[1].get_level_values(0) if any([True for part in ['cnae','ocup'] if part in
col])]
occup_cols_no_rep = list(set(occup_cols))

# Columns to be removed
print(occup_cols_no_rep)

# Dropping occupation columns
df_no_occup = df.drop(occup_cols_no_rep, axis=1, level=0)
print(df_no_occup.shape)
df_no_occup.head()
```

```
['te_res_ocup_pret', 'te_nat_cnae', 'te_res_cnae', 'te_nat_instrucao_ocup_pret', 'te_res_ocup_atual',
'te_nat_sexo_cnae', 'te_nat_ocup_atual', 'te_nat_ocup_pret']
(753, 69)
```

| cd_indicador | te_insp_rgt | te_inspecoes | te_nat | te_nat_idade | te_nat_instrucao | ... | te_res_raca | te_res_raca_idade | te |
|---|---|---|---|---|---|---|---|---|---|

| ds_agreg_primaria | default | default | default | adult | default | minor | 5º Ano Completo | 6º ao 9º Ano Incompl | >Ignorado | Analfabeto | ... | Pessoa Que Se Enquadrar Como de Raça Amarela ( de Origem Japonesa, Chinesa, Coreana, Etc) | >Não Informado | Pessoa Que Se Enquadrar Como Branca | Pessoa Que Se Enquadrar Como Parda ou Se Declarar Como Mulata, Cabocla, Cafuza, Mameluca ou Mestiça de Preto com Pessoa de Outra Cor ou Raça | Pessoa Que Se Enquadrar Como Preta | Pessoa Que Se Enquadrar Como de Raça Amarela ( de Origem Japonesa, Chinesa, Coreana, Etc) | Fe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **6** | 10 | 16 | 27 | 22 | 0 | 5 | 0 | 3 | 0 | 11 | ... | 2 | 0 | 0 | 0 | 0 | 0 | 5 |
| **11** | 3 | 3 | 6 | 6 | 0 | 0 | 0 | 1 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **20** | 1 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **22** | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| **24** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 69 columns

After dropping the occupations data, 60 features remained in the dataset.

From that collection, some actually have a secondary aggregation feature, not present in the CSV. Those will be removed.

```
# Getting the remaining indicators' codes
print(df_no_occup.axes[1].get_level_values(0))

# Indicators known (business logic) to have a secondary level
cols_sec_level = ['te_nat_instrucao_idade', 'te_nat_raca_idade', 'te_res_instrucao_idade', 'te_res_raca_idade']
df_final = df_no_occup.drop(cols_sec_level, axis=1, level=0)

print(df_final.shape)

# Checking sparsity/density
sdf = df.to_sparse()
print(sdf.density)

df_final.head()
```

```
Index(['te_insp_rgt', 'te_inspecoes', 'te_nat', 'te_nat_idade', 'te_nat_idade',
       'te_nat_idade', 'te_nat_instrucao', 'te_nat_instrucao',
       'te_nat_instrucao', 'te_nat_instrucao', 'te_nat_instrucao',
       'te_nat_instrucao', 'te_nat_instrucao', 'te_nat_instrucao',
       'te_nat_instrucao', 'te_nat_instrucao', 'te_nat_instrucao',
       'te_nat_instrucao_idade', 'te_nat_instrucao_idade',
       'te_nat_instrucao_idade', 'te_nat_instrucao_idade', 'te_nat_raca',
       'te_nat_raca', 'te_nat_raca', 'te_nat_raca', 'te_nat_raca',
       'te_nat_raca', 'te_nat_raca_idade', 'te_nat_raca_idade',
       'te_nat_raca_idade', 'te_nat_raca_idade', 'te_nat_raca_idade',
       'te_nat_sexo', 'te_nat_sexo', 'te_ope', 'te_res', 'te_res_idade',
       'te_res_idade', 'te_res_idade', 'te_res_instrucao', 'te_res_instrucao',
       'te_res_instrucao', 'te_res_instrucao', 'te_res_instrucao',
       'te_res_instrucao', 'te_res_instrucao', 'te_res_instrucao',
       'te_res_instrucao', 'te_res_instrucao', 'te_res_instrucao',
       'te_res_instrucao_idade', 'te_res_instrucao_idade',
       'te_res_instrucao_idade', 'te_res_instrucao_idade', 'te_res_raca',
       'te_res_raca', 'te_res_raca', 'te_res_raca', 'te_res_raca',
       'te_res_raca', 'te_res_raca_idade', 'te_res_raca_idade',
       'te_res_raca_idade', 'te_res_raca_idade', 'te_res_raca_idade',
       'te_res_sexo', 'te_res_sexo', 'te_rgt', 'te_rgt_per_insp'],
      dtype='object', name='cd_indicador')
```

```
(753, 51)
1.0
```

| cd_indicador | te_insp_rgt | te_inspecoes | te_nat | te_nat_idade | | | te_nat_instrucao | | | | ... | te_res_raca | | | | | | te_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ds_agreg_primaria | default | default | default | adult | default | minor | 5º Ano Completo | 6º ao 9º Ano Incompl | >Ignorado | Analfabeto | ... | >Não Informado | Pessoa Que Se Enquadrar Como Branca | Pessoa Que Se Enquadrar Como Indígena ou Índia | Pessoa Que Se Enquadrar Como Parda ou Se Declarar Como Mulata, Cabocla, Cafuza, Mameluca ou Mestiça de Preto com Pessoa de Outra Cor ou Raça | Pessoa Que Se Enquadrar Como Preta | Pessoa Que Se Enquadrar Como de Raça Amarela ( de Origem Japonesa, Chinesa, Coreana, Etc) | Fer |
| **6** | 10 | 16 | 27 | 22 | 0 | 5 | 0 | 3 | 0 | 11 | ... | 63 | 1 | 0 | 10 | 1 | 2 | 5 |
| **11** | 3 | 3 | 6 | 6 | 0 | 0 | 0 | 1 | 0 | 1 | ... | 4 | 0 | 0 | 0 | 1 | 0 | 0 |
| **20** | 1 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **22** | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 1 | 2 | 0 | 6 | 2 | 5 | 0 |
| **24** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 51 columns

The final dataset has 51 features.

```
from sklearn.preprocessing import MinMaxScaler

# Initialize a scaler, then apply it to the features
scaler = MinMaxScaler() # default=(0, 1)
feats = df_final.axes[1].get_level_values(0)[:-1]

df_final = pd.DataFrame(data = df_final)
df_final.loc[idx[:], idx[feats,:]] = scaler.fit_transform(df_final.loc[idx[:], idx[feats,:]])

df_final.head()
```

| cd_indicador | te_insp_rgt | te_inspecoes | te_nat | te_nat_idade | | | te_nat_instrucao | | | | ... | te_res_raca | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ds_agreg_primaria | default | default | default | adult | default | minor | 5º Ano Completo | 6º ao 9º Ano Incompl | >Ignorado | Analfabeto | ... | >Não Informado | Pessoa Que Se Enquadrar Como Branca | Pessoa Que Se Enquadrar Como Indígena ou Índia | Pessoa Que Se Enquadrar Como Parda ou Se Declarar Como Mulata, Cabocla, Cafuza, Mameluca ou Mestiça de Preto com Pessoa de Outra Cor ou Raça | Pessoa Que Se Enquadrar Como Preta | Pessoa Que Se Enquadrar Como de Raça Amarela ( de Origem Japonesa, Chinesa, Coreana, Etc) |
| **6** | 0.120000 | 0.142857 | 0.056250 | 0.046025 | 0.0 | 0.357143 | 0.0 | 0.061224 | 0.0 | 0.026442 | ... | 0.166227 | 0.014085 | 0.0 | 0.089286 | 0.023810 | 0.027397 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **11** | 0.026667 | 0.019048 | 0.012500 | 0.012552 | 0.0 | 0.000000 | 0.0 | 0.020408 | 0.0 | 0.002404 | ... | 0.010554 | 0.000000 | 0.0 | 0.000000 | 0.023810 | 0.000000 |
| **20** | 0.000000 | 0.000000 | 0.004167 | 0.004184 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.002639 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 |
| **22** | 0.013333 | 0.009524 | 0.004167 | 0.004184 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.002639 | 0.028169 | 0.0 | 0.053571 | 0.047619 | 0.068493 |
| **24** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 |

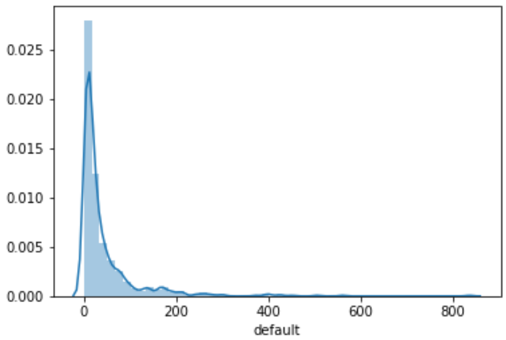
5 rows × 51 columns

### Setting the labeling class

As proposed, the labeling class will be based on the rescues per inspection (te_rgt_per_insp). The labels (LOW, MEDIUM and HIGH) will be based on the terciles of its distribution.

```
import seaborn as sns
sns.distplot(df_final.te_rgt_per_insp.default)

df_final.te_rgt_per_insp.default.describe()
```

```
count    753.000000
mean      43.193323
std       72.725993
min        0.250000
25%        8.000000
50%       18.000000
75%       45.000000
max      836.500000
Name: default, dtype: float64
```




As the distribution shows, there's a concentration of rescues per inspection near the origin. It could mean few occurences of modern slavery, if there weren't municipalities that presented much higher rates - leading to a long tail in the distribution. Perfecting resources placement could lower inspections with near-zero rescues and bring the distribution to a normal display. At the same time, incresing effectiveness in enforcement may reduce the cases with the highest rates.

```
# Categorization of the label feature
df_with_labels = df_final
df_with_labels.loc[idx[:], idx['te_rgt_per_insp',:]] = pd.qcut(
    df_final.te_rgt_per_insp.default,
    3,
    labels=["LOW", "MEDIUM", "HIGH"]
)

print(df_with_labels.te_rgt_per_insp.default.describe())

print('LOW MAX: ' + str(
    df_with_labels.loc[df_with_labels.te_rgt_per_insp.default == 'LOW', idx['te_rgt',:]].values.max())
)
print('MEDIUM MAX: ' + str(
    df_with_labels.loc[df_with_labels.te_rgt_per_insp.default == 'MEDIUM', idx['te_rgt',:]].values.max())
)

df_with_labels.head()
```

```
count      753
unique       3
top        LOW
freq       255
```

```
Name: default, dtype: object
LOW MAX: 0.04525862068965517
MEDIUM MAX: 0.13864942528735633
```

| cd_indicador | te_insp_rgt | te_inspecoes | te_nat | te_nat_idade | | | te_nat_instrucao | | | | ... | te_res_raca | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ds_agreg_primaria | default | default | default | adult | default | minor | 5º Ano Completo | 6º ao 9º Ano Incompl | >Ignorado | Analfabeto | ... | >Não Informado | Pessoa Que Se Enquadrar Como Branca | Pessoa Que Se Enquadrar Como Indígena ou Índia | Pessoa Que Se Enquadrar Como Parda ou Se Declarar Como Mulata, Cabocla, Cafuza, Mameluca ou Mestiça de Preto com Pessoa de Outra Cor ou Raça | Pessoa Que Se Enquadrar Como Preta | Pessoa Que Se Enquadrar Como de Raça Amarela ( de Origem Japonesa, Chinesa, Coreana, Etc) |
| **6** | 0.120000 | 0.142857 | 0.056250 | 0.046025 | 0.0 | 0.357143 | 0.0 | 0.061224 | 0.0 | 0.026442 | ... | 0.166227 | 0.014085 | 0.0 | 0.089286 | 0.023810 | 0.027397 |
| **11** | 0.026667 | 0.019048 | 0.012500 | 0.012552 | 0.0 | 0.000000 | 0.0 | 0.020408 | 0.0 | 0.002404 | ... | 0.010554 | 0.000000 | 0.0 | 0.000000 | 0.023810 | 0.000000 |
| **20** | 0.000000 | 0.000000 | 0.004167 | 0.004184 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.002639 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 |
| **22** | 0.013333 | 0.009524 | 0.004167 | 0.004184 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.002639 | 0.028169 | 0.0 | 0.053571 | 0.047619 | 0.068493 |
| **24** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.000000 | ... | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.000000 | 0.000000 |

5 rows × 51 columns

### Implementation

As proposed, the dataset will be split in two: a 50% subset for training and validating the random forest (in a proportion of 80-20) and the other 50% will be used for testing.

**Splitting the dataset**

```
from sklearn.model_selection import train_test_split

rs = 64

# Split the features and labeling class in train, validate and test
feats = df_final.axes[1].get_level_values(0)[:-1]
labels = df_final.axes[1].get_level_values(0)[-1]

feats_subset = df_final.loc[idx[:], idx[feats,:]]
labels_subset = df_final.loc[idx[:], idx[labels,:]]

# Splitting in half for training/validation and testing
X_trainvalidate, X_test, y_trainvalidate, y_test = train_test_split(
    feats_subset, labels_subset, test_size = 0.5, random_state = rs
)

# Splitting the train/validate into training and validating subsets
X_train, X_validate, y_train, y_validate = train_test_split(
    X_trainvalidate, y_trainvalidate, test_size = 0.2, random_state = rs
)

print(X_train.shape)
print(y_train.shape)
print(X_validate.shape)
print(y_validate.shape)
print(X_test.shape)
print(y_test.shape)
```

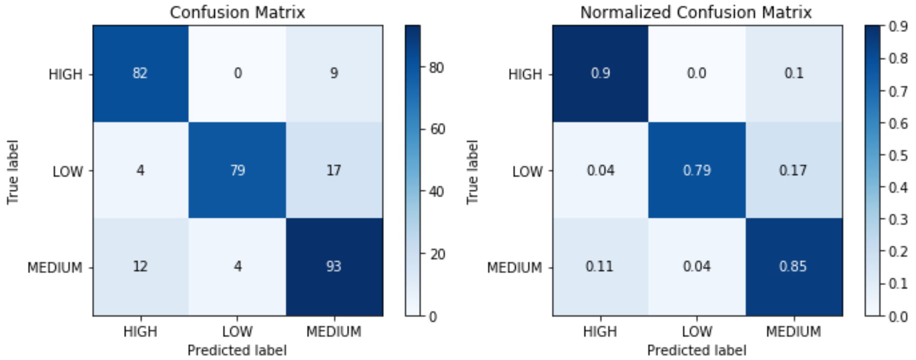
```
(300, 50)
(300, 1)
```

```
(76, 50)
(76, 1)
(377, 50)
(377, 1)
```

**Training the model**

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_predict

classifier = RandomForestClassifier(n_estimators=5, max_depth=5, random_state=rs)
learner = classifier.fit(X_train, y_train.values.ravel())
predictions_train = cross_val_predict(learner, X_train, y_train.values.ravel(), cv=10)

# Metrics
from sklearn.metrics import fbeta_score, accuracy_score

print('Accuracy: ' + str(accuracy_score(y_train, predictions_train)))
print('F Score (each label): ' + str(fbeta_score(y_train, predictions_train, beta = 0.5, average=None)))
print('F Score: ' + str(fbeta_score(y_train, predictions_train, beta = 0.5, average='micro')))

# Confusion matrix
import matplotlib.pyplot as plt
import scikitplot as skplt

skplt.metrics.plot_confusion_matrix(y_train, predictions_train, normalize=False)
skplt.metrics.plot_confusion_matrix(y_train, predictions_train, normalize=True)
plt.show()
```

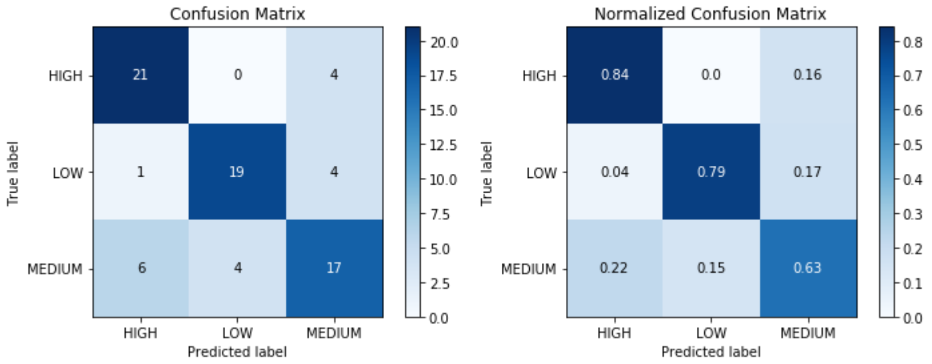
```
Accuracy: 0.8466666666666667
F Score (each label): [0.84886128 0.91435185 0.79487179]
F Score: 0.8466666666666667
```




**Validating the model**

```python
predictions_validate = cross_val_predict(learner, X_validate, y_validate.values.ravel(), cv=10)

# Metrics
print('Accuracy: ' + str(accuracy_score(y_validate, predictions_validate)))
print('F Score (each label): ' + str(fbeta_score(y_validate, predictions_validate, beta = 0.5, average=None)))
print('F Score: ' + str(fbeta_score(y_validate, predictions_validate, beta = 0.5, average='micro')))

# Confusion matrix
skplt.metrics.plot_confusion_matrix(y_validate, predictions_validate, normalize=False)
skplt.metrics.plot_confusion_matrix(y_validate, predictions_validate, normalize=True)
plt.show()
```

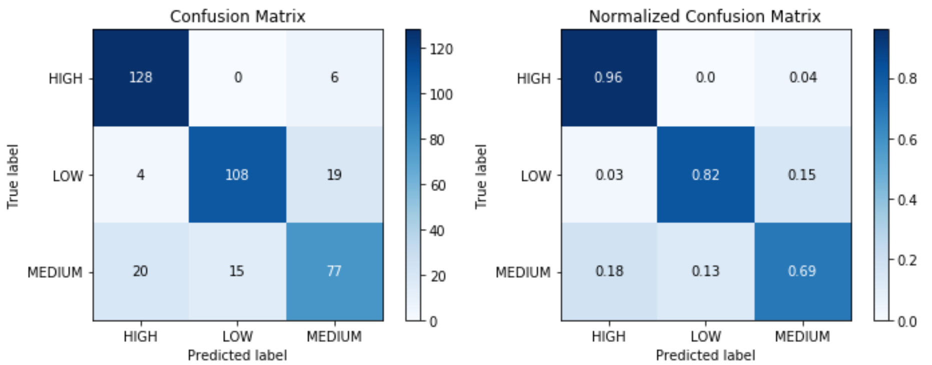
```
Accuracy: 0.75
F Score (each label): [0.76642336 0.81896552 0.66929134]
F Score: 0.75
```




#### Testing the model

```
predictions_test = cross_val_predict(learner, X_test, y_test.values.ravel(), cv=10)

# Metrics
print('Accuracy: ' + str(accuracy_score(y_test, predictions_test)))
print('F Score (each label): ' + str(fbeta_score(y_test, predictions_test, beta = 0.5, average=None)))
print('F Score: ' + str(fbeta_score(y_test, predictions_test, beta = 0.5, average='micro')))

# Confusion matrix
skplt.metrics.plot_confusion_matrix(y_test, predictions_test, normalize=False)
skplt.metrics.plot_confusion_matrix(y_test, predictions_test, normalize=True)
plt.show()
```

```
Accuracy: 0.830238726790451
F Score (each label): [0.86253369 0.86677368 0.74038462]
F Score: 0.830238726790451
```




### Refinement

Starting from the parameters n_estimators = 10 and max_depth = 10, we started tuning them to acheieve better metrics. The best numbers were found with both parameters in 5.

#### Starting metrics for first hyperparameters setup

n_estimators = 10 max_depth = 10 Accuracy: 0.7833333333333333 F Score (each label): [0.78833693 0.83333333 0.73608618] F Score: 0.7833333333333333

#### Best configuration

n_estimators = 5 max_depth = 5 Accuracy: 0.8466666666666667 F Score (each label): [0.84886128 0.91435185 0.79487179] F Score: 0.8466666666666667

Although the increase was not observed in the smaller validation subset, it resurfaced in the validation:

#### Validation - Starting metrics for first hyperparameters setup

n_estimators = 10 max_depth = 10 Accuracy: 0.7631578947368421 F Score (each label): [0.7751938 0.76612903 0.7480315 ] F Score: 0.7631578947368421

#### Validation - Best configuration

n_estimators = 5 max_depth = 5 Accuracy: 0.75 F Score (each label): [0.76642336 0.81896552 0.66929134] F Score: 0.75

#### Test - Starting metrics for first hyperparameters setup

n_estimators = 10 max_depth = 10 Accuracy: 0.8169761273209549 F Score (each label): [0.85149864 0.83710407 0.73770492] F Score: 0.8169761273209549

**Test - Best configuration**

n_estimators = 5 max_depth = 5 Accuracy: 0.830238726790451 F Score (each label): [0.86253369 0.86677368 0.74038462] F Score: 0.830238726790451

## IV. Results

### Model Evaluation and Validation

With hyperparameters adjusted over the cross-validated random forest, the results are reliable and robust, therefore it can be put to use in flagging municipalities for maximizing rescues per inspection. Splitting the dataset in different sizes allow us to observe that different data may result in little variation on accuracy and f-score, but the model is still reliable even with such perturbation.

### Justification

As expected, the larger the balanced subset, the more accurate the cross-validated prediction. The testing subset, that held 50% of the data had the best metrics overall (except for the MEDIUM label).

In the proposed real-world scenario, though, the accuracy can only be assessed by addressing resources to the flagged municipalities (labeled HIGH according to an expectation of maximizing rescues per inspection). It's very promising, nonetheless, to run such experimentations: in the normalized confusion matrix, for instance, we can see that no occurences of real HIGH were misclassified as LOW and 0.04 (6 instances) as MEDIUM, while 0.96 (128 municipalities) were correctly classified.

## V. Conclusion

### Free-Form Visualization

Resource allocation for maximizing the number rescues from modern slavery per inspections can be achieved by changing from the denounce-based traditional approach to a evidence-based classification of municipalities, tackling the problem by using statistical analysis to increase the probability of more effective diligences. One way to do that is classifying municipalities by their profile, along with historical rescues data - informations already available to goverment agencies.

The resulting model is accurate, robust and reliable. If used in an iterative way, with feedback loops, the model will be constantly adapted to the trends in modern slavery.

### Reflection

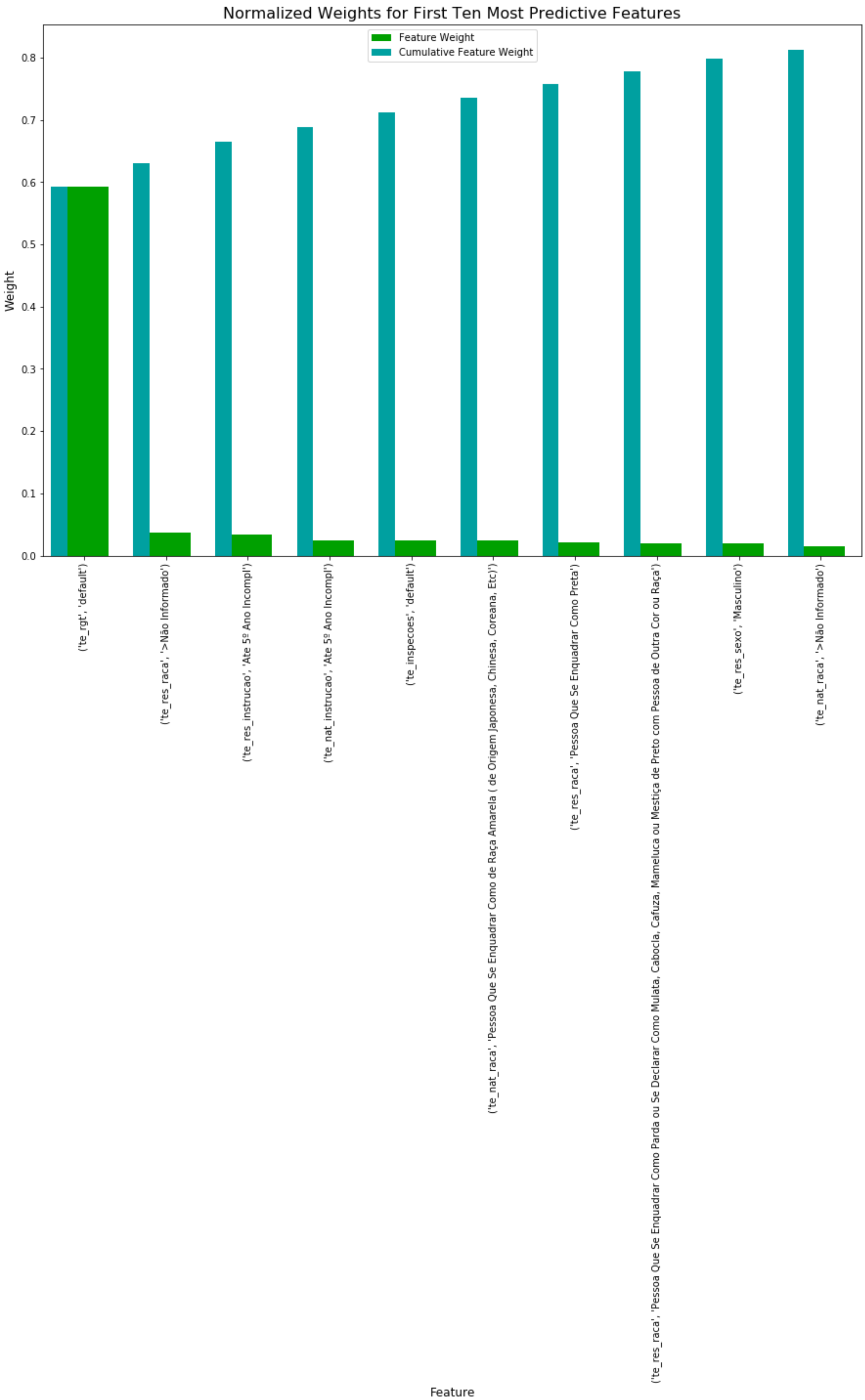
The data from the census were explored and, due to sparsity, discarded in favor of using only indicators generated by the rescues. The high dimensionality were reduced first by observing business rules (like absence of higher granularity data - ds_agregacao_secundaria) and collections of features too granular, that rendered the dataset too sparse. We ended up with 51 features. One of them, the rescues per inspection indicator, were transformed in a categorical field, labeling the instance as HIGH, MEDIUM or LOW rescues per inspection, using the terciles of the distribution of the original feature.

The resulting dataset was subject to a cross-validated Random Forests - ensemble algorithm for classification. Its hyperparameters tuned for increasing accuracy and f-score metrics in the training subset. The result was over 83% accuracy in the testing subset, with a similar f-score.

If law enforcement should address the 150 municipalities classified as HIGH, we should expect the same 128 places (~85.33%) to be correctly classified, therefore generating more rescues per inspection. 4 of them (~2.67%) would represent a low rate of success, since the municipalities labeled as LOW are near-zero rescues per inspections, from the distribution used for labeling (from 0 to ~0.045). Similarly, we'd have 20 inspections (~13.33% of the 150 labeled as HIGH) with fewer rescues (from ~0.045 to ~0.139) according to the ratings and the confusion matrix.

We can also verify the actual feature importance for the accurate predictions, witch may enable us to avoid rhetorical disputes and focus on tackling the problem at hand.

```
# Display the most important features
import numpy as np

importances = learner.feature_importances_

indices = np.argsort(importances)[::-1]
columns = X_test.columns.values[indices[:10]]
values = importances[indices][:10]

# Create the plot
fig = plt.figure(figsize = (15,10))
plt.title("Normalized Weights for First Ten Most Predictive Features", fontsize = 16)
plt.bar(np.arange(10), values, width = 0.6, align="center", color = '#00A000', \
      label = "Feature Weight")
plt.bar(np.arange(10) - 0.3, np.cumsum(values), width = 0.2, align = "center", color = '#00A0A0', \
      label = "Cumulative Feature Weight")
plt.xticks(np.arange(10), columns, rotation='vertical')
plt.xlim((-0.5, 9.5))
plt.ylabel("Weight", fontsize = 12)
plt.xlabel("Feature", fontsize = 12)

plt.legend(loc = 'upper center')
plt.tight_layout()
plt.show()
```




Since the dataset used is closely related to inspections and rescues, it tends to a reinforced bias (the number of rescues - te_rgt -, for instance, is obviously extremely relevant to the classification. Some relevant features can bee seen, such as the ammount Male slaves rescued (te_res_sexo - Masculino), the low instruction level (te_res_instrucao and te_nat_instrucao - 5th and 6th to 9th year of formal education). Some issues show up also, such as the race not informed in the reports (te_nat_raca and te_res_raca) being among the most relevant features.

### Improvement

Since the results of this study is expected to be included in the Observatório Digital do Trabalho Escravo, the model will be built using the full inspections dataset. Then, it will be applied for labeling municipalities with no inspection record. In order for that to work, we must revisit the raw datasets from IBGE and the check if the high sparsity persists and, if so, how to address this issue.

Those municipalities, now identified, with a label HIGH will be flagged, as the ones already identified in the inspections records. The final output would be a brazilian map of municipalities colored by rating level, revealing which areas should be subject to a more thorough investigation.