UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

RELATÓRIO DA APLICAÇÃO DE MINERAÇÃO DE DADOS E TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA CRIAÇÃO DE UM MODELO DE PREDIÇÃO DO NÚMERO DE ACIDENTES DE TRÂNSITO NO ESTADO DO CEARÁ

DISCIPLINA: MINERAÇÃO MASSIVA DE DADOS

FULANO DE TAL

FORTALEZA–CEARÁ

20xx

Introdução

O objetivo desse relatório é descrever o processo de desenvolvimento de um modelo de predição do número de acidentes de trânsito no estado do Ceará utilizando técnicas de mineração de dados e aprendizado de máquina. Esse experimento é resultado dos conceitos aprendidos na disciplina de Mineração Massiva de Dados, bem como estudo do material disponibilizado para pesquisa.

O relatório contém a contextualização do problema, o desenvolvimento do experimento e suas etapas – com uma breve explanação sobre as técnicas utilizadas e resultados obtidos, e uma conclusão sobre o experimento.

Predição do número de acidentes de trânsito no estado do Ceará

Um modelo preditivo é importante para a gestão pública pelo fato de auxiliar nas decisões estratégicas de governo, de maneira que o gestor possa se preparar para o futuro e evitar situações indesejadas. O modelo se baseia em dados reais para criar uma equação matemática, baseada nas entradas escolhidas (*features*), que irá prever uma variável de saída – no caso desse trabalho, o número de acidentes de trânsito.

$y = f(x)$, onde y é a predição e x são as *features*.

Apesar de intuitivamente modelos preditivos serem aplicados apenas para a adequada preparação dos interessados no enfrentamento de cenários futuros inexoráveis, o modelo proposto também se destaca, em termos de entrega de valor, por sua capacidade de avaliação de possíveis cenários futuros que podem ser simulados para tomada de decisões. Nesse sentido, o gestor poderá, com base em *features* de entrada simuladas e no resultado apresentado pelo modelo para tais *features*, obter indicações de como estas entradas devem ser ajustadas para se alcançar o futuro desejado, que no caso é a diminuição dos índices de acidentes de trânsito. De posse das *features* que devem ser ajustadas e do quanto tal ajuste deve ser feito, o gestor se empodera de um mapa de gestão que possibilita a construção de um planejamento mais científico e com muito mais possibilidade de sucesso. Assim, o modelo proposto se converte em uma interessante ferramenta de apoio a gestão, que usa o modelo preditivo para tomadas de decisões presentes que tenham real impacto sobre os indicadores de acidentalidade no futuro.

A abordagem utilizada nesse experimento foi uma **regressão**, utilizando o algoritmo ***Randon Forest***.

Desenvolvimento

Com o intuito de prover uma melhor organização do experimento, bem como otimizar o fluxo de trabalho, tornando-o simples, veloz, portável e reutilizável, o fluxo de trabalho foi dividido em 3 etapas distintas: *Feature Engineering*, *Grid Search & Cross Validation* e *Training & Test*. Essa divisão por etapas de aprendizado de máquina facilitou a reexecução destas, conforme o experimento era realizado e o modelo era ajustado e testado.

Para o experimento foi utilizado o Jupyter Notebook com Python 3.

Feature Engineering

A técnica de *Feature Engineering* aumenta o poder preditivo dos algoritmos de aprendizado de máquina, a partir dos recursos dos dados. Para esse experimento, no âmbito dessa técnica, foram realizadas as seguintes tarefas:

1. Escolha dos melhores atributos;
2. Criação das *features*;
3. Geração dos *datasets*;
4. Limpeza dos dados (remoção dos valores nulos dos *datasets*);
5. Verificação do grau de correlação das variáveis (feature) com a saída;
6. Escolha das melhores *features* com base nas correlações.

Inicialmente, realizou-se um estudo sobre os dados públicos disponíveis na web, que intuitivamente pudessem ter alguma relação com a acidentalidade de trânsito. Dessa maneira, cada base coletada se encaixava em um determinado **aspecto** que poderia ter influência sobre o número de acidentes de trânsito no Ceará. Abaixo são listados os aspectos analisados, os *datasets* gerados a partir dos dados e sua respectiva fonte.

a) Aspectos dos Acidentes

Dataset:	Acidentes_2014-2017.csv
Features:	MÊS
	TOTAL_FROTA_ACUMULADO
	TOTAL_FROTA_MOTOCICLETA_ACUMULADO
	TOTAL_ACIDENTES_MES
	TOTAL_ACIDENTES_ANO_ANTERIOR
	MEDIA_MENSAL_ACIDENTES_ANO_ANTERIOR
	PERCENTUAL_ACIDENTES_FROTA_ACUMULADO
	TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR
	MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR
	TOTAL_FERIDOS_ACIDENTE_ANO_ANTERIOR
	TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR
Fonte:	http://portal.detran.ce.gov.br/index.php/estatisticas/estatisticas-de-acidentes (acesso em 15/12/2018)
	https://www.denatran.gov.br/estatistica/237-frota-veiculos (acesso em 16/12/2018).

b) Aspectos Econômicos

Dataset:	Arrecadacao_IRRF_Trabalho.csv
Features:	MÊS
	VALOR_ARRECADADO_IRRF_TRABALHO
Fonte:	http://idg.receita.fazenda.gov.br/dados/receitadata/arrecadacao/arrecadacao-por-estado (acesso em 16/12/2018)

Dataset:	DadosIPCAeINPC_2015-2018_IBGE.csv
Features:	MES
	BRASIL_IPCA_VARIACAO_MENSAL
	BRASIL_IPCA_VARIACAO_12MESES
	METROFORTALEZA_IPCA_VARIACAO_MENSAL
	METROFORTALEZA_IPCA_VARIACAO_12MESES
	BRASIL_INPC_VARIACAO_MENSAL
	BRASIL_INPC_VARIACAO_12MESES
	METROFORTALEZA_INPC_VARIACAO_MENSAL
	METROFORTALEZA_INPC_VARIACAO_12MESES

Fonte:	https://sidra.ibge.gov.br/tabela/1419 (acesso em 16/12/2018)

c) Aspectos da Frota

Dataset:	FrotaCeara2015-2018_Denatran.csv
Features:	MÊS
	TOTAL_FROTA_ACUMULADO
	TOTAL_FROTA_DIFERENCA_MES
	TOTAL_TIPO_AUTOMOVEL
	TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES
	TOTAL_TIPO_CAMINHAO
	TOTAL_TIPO_CAMINHAO_DIFERENCA_MES
	TOTAL_TIPO_CAMINHÃO_TRATOR
	TOTAL_TIPO_CAMINHONETE
	TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES
	TOTAL_TIPO_CAMIONETA
	TOTAL_TIPO_CAMIONETA_DIFERENCA_MES
	TOTAL_TIPO_CHASSI_PLATAFORMA
	TOTAL_TIPO_CICLOMOTOR
	TOTAL_TIPO_MICROÔNIBUS
	TOTAL_TIPO_MOTOCICLETA
	TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES
	TOTAL_TIPO_MOTONETA
	TOTAL_TIPO_MOTONETA_DIFERENCA_MES
	TOTAL_TIPO_ÔNIBUS
	TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES
	TOTAL_TIPO_REBOQUE
	TOTAL_TIPO_SEMI_REBOQUE
	TOTAL_TIPO_REBOQUE_DIFERENCA_MES
	TOTAL_TIPO_SIDE_CAR
	TOTAL_TIPO_OUTROS
	TOTAL_TIPO_TRATOR_RODAS
	TOTAL_TIPO_TRICICLO
	TOTAL_TIPO_UTILITÁRIO
	TOTAL_TIPO_UTILITÁRIO_DIFERENCA_MES
Fonte:	https://www.denatran.gov.br/estatistica/237-frota-veiculos (acesso em 16/12/2018). (*Os dados dizem respeito a frota acumulada no mês.

Vale ressaltar que outros aspectos, intuitivamente, também seriam interessantes, como os “Aspectos do Condutor” (escolaridade, quantidade de multas etc.), no entanto estes dados não estavam disponíveis na web.

Os dados de cada *dataset* foram analisados e unificados em um único dataset, contendo dados mês a mês, desde janeiro de 2015 até dezembro de 2017 (**número de linhas do *dataset* = 36**). Nessa unificação, foram descartados dados que se repetiam em diferentes datasets (TOTAL_FROTA_ANO), bem como *features* vazias. A seguir é mostrada a estrutura do *dataset* unificado:

Dataset:	Acidentalidade_DETRAN_CE_2015-2017.csv
Features:	**MES (ANO/MÊS de referência)**
	TOTAL_ACIDENTES_MES (variável Y – saída que se deseja prever)
	TOTAL_ACIDENTES_ANO_ANTERIOR
	MEDIA_MENSAL_ACIDENTES_ANO_ANTERIOR
	PERCENTUAL_ACIDENTES_FROTA_ACUMULADO
	TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR
	MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR
	TOTAL_FERIDOS_ACIDENTE_ANO_ANTERIOR
	TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR
	VALOR_ARRECADADO_IRRF_TRABALHO
	BRASIL_IPCA_VARIACAO_MENSAL
	BRASIL_IPCA_VARIACAO_12MESES
	METROFORTALEZA_IPCA_VARIACAO_MENSAL
	METROFORTALEZA_IPCA_VARIACAO_12MESES
	BRASIL_INPC_VARIACAO_MENSAL

	BRASIL_INPC_VARIACAO_12MESES
	METROFORTALEZA_INPC_VARIACAO_MENSAL
	METROFORTALEZA_INPC_VARIACAO_12MESES
	TOTAL_FROTA_ACUMULADO
	TOTAL_FROTA_DIFERENCA_MES
	TOTAL_TIPO_AUTOMOVEL
	TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES
	TOTAL_TIPO_CAMINHAO
	TOTAL_TIPO_CAMINHAO_DIFERENCA_MES
	TOTAL_TIPO_CAMINHÃO_TRATOR
	TOTAL_TIPO_CAMINHONETE
	TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES
	TOTAL_TIPO_CAMIONETA
	TOTAL_TIPO_CAMIONETA_DIFERENCA_MES
	TOTAL_TIPO_CHASSI_PLATAFORMA
	TOTAL_TIPO_CICLOMOTOR
	TOTAL_TIPO_MICROÔNIBUS
	TOTAL_TIPO_MOTOCICLETA
	TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES
	TOTAL_TIPO_MOTONETA
	TOTAL_TIPO_MOTONETA_DIFERENCA_MES
	TOTAL_TIPO_ÔNIBUS
	TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES
	TOTAL_TIPO_REBOQUE
	TOTAL_TIPO_SEMI_REBOQUE
	TOTAL_TIPO_REBOQUE_DIFERENCA_MES
	TOTAL_TIPO_SIDE_CAR
	TOTAL_TIPO_OUTROS
	TOTAL_TIPO_TRATOR_RODAS
	TOTAL_TIPO_TRICICLO
	TOTAL_TIPO_UTILITÁRIO
	TOTAL_TIPO_UTILITÁRIO_DIFERENCA_MES

Após a importação do arquivo .csv, a variável de saída **y** (TOTAL_ ACIDENTES_MES) da amostra, teve-se os seguintes dados estatísticos:

```
In [2]: import numpy as np
        import pandas as pd
        from scipy import stats
        from scipy.stats import shapiro
        import matplotlib
        import matplotlib.pyplot as plt

        ## Importa o csv com os dados de acidentes
        data = pd.read_csv('Acidentalidade_DETRAN_CE_2015-2017.csv')

        ## Descreve estatisticamente cada coluna do dataset importado
        data.describe()
```

Out[2]:

	TOTAL_ACIDENTES_MES	TOTAL_ACIDENTES_ANO_ANTERIOR	MEDIA_MENSAL_ACIDENTES_ANO_ANTERIOR	PERCENTUAL_ACIDENTES_FROTA_ACI
count	36.000000	36.000000	36.000000	
mean	2425.222222	29151.333333	2429.000000	
std	151.452041	1098.923837	91.494282	
min	2160.000000	27900.000000	2325.000000	
25%	2330.000000	27900.000000	2325.000000	
50%	2437.500000	29011.000000	2417.000000	
75%	2515.250000	30543.000000	2545.000000	
max	2793.000000	30543.000000	2545.000000	

8 rows × 46 columns

Sobre o *dataset* final foram então realizados os **testes de normalidade**. Na Estatística, esses testes são utilizados para verificar se a distribuição de probabilidade associada a um conjunto de dados pode ser aproximada pela distribuição normal. O resultado deles tem implicações na forma de análise das variáveis do *dataset*. Para variáveis que seguem normalidade a análise é feita com base na inferência estatística, para dados não normais são usadas técnicas de estatística não paramétrica para extrair informações do banco.

Para testar a normalidade do banco utilizou-se o *Shapiro Wilk Test*. O propósito do *Shapiro Wilk Test* é fornecer uma estatística de teste para avaliar se uma amostra tem **distribuição normal**. O resultado do teste mostrou um ***p-value* = 0.00**, que mostra que os dados **não têm uma distribuição normal**.

```
In [2]: ## Executa o teste de normalidade dos dados
        stats.shapiro(data)

Out[2]: (0.13855212926864624, 0.0)
```

Feito isso, foram analisadas as correlações da **variável y** com as **varáveis x.** Para ver as correlações entre as variáveis, foi utilizado o método “Pearson” (padrão no **pandas.DataFrame.corr**). Nessa análise verificou-se que **todas as correlações estavam baixas**.

```
In [3]: ## Verifica as correlações entre as variáveis do dataset de acidentes usando o método de "Pearson".
        data_corr = data.corr()

        ## Exibe as correlações com total_acidentes_mes
        data_corr['TOTAL_ACIDENTES_MES']

Out[3]: TOTAL_ACIDENTES_MES                          1.000000
        TOTAL_ACIDENTES_ANO_ANTERIOR                -0.270967
        MEDIA_MENSAL_ACIDENTES_ANO_ANTERIOR         -0.272325
        PERCENTUAL_ACIDENTES_FROTA_ACUMULADO         0.806794
        TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR    -0.428317
        MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR -0.425051
        TOTAL_FERIDOS_ACIDENTE_ANO_ANTERIOR          0.014095
        TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR           0.389029
        VALOR_ARRECADADO_IRRF_TRABALHO              -0.258716
        BRASIL_IPCA_VARIACAO_MENSAL                  0.166793
        BRASIL_IPCA_VARIACAO_12MESES                 0.144695
        METROFORTALEZA_IPCA_VARIACAO_MENSAL          0.318803
        METROFORTALEZA_IPCA_VARIACAO_12MESES        -0.016017
        BRASIL_INPC_VARIACAO_MENSAL                  0.193755
        BRASIL_INPC_VARIACAO_12MESES                 0.123380
        METROFORTALEZA_INPC_VARIACAO_MENSAL          0.250032
        METROFORTALEZA_INPC_VARIACAO_12MESES        -0.060407
        TOTAL_FROTA_ACUMULADO                       -0.321841
        TOTAL_FROTA_DIFERENCA_MES                    0.448403
        TOTAL_TIPO_AUTOMOVEL                        -0.315054
        TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES           0.508543
        TOTAL_TIPO_CAMINHAO                         -0.331548
        TOTAL_TIPO_CAMINHAO_DIFERENCA_MES            0.251740
        TOTAL_TIPO_CAMINHÃO_TRATOR                  -0.295861
        TOTAL_TIPO_CAMINHONETE                      -0.313009
        TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES         0.395721
        TOTAL_TIPO_CAMIONETA                        -0.323238
        TOTAL_TIPO_CAMIONETA_DIFERENCA_MES           0.444738
        TOTAL_TIPO_CHASSI_PLATAFORMA                 0.153604
        TOTAL_TIPO_CICLOMOTOR                       -0.186100
        TOTAL_TIPO_MICROÔNIBUS                      -0.309308
        TOTAL_TIPO_MOTOCICLETA                      -0.332052
        TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES         0.448273
        TOTAL_TIPO_MOTONETA                         -0.320630
        TOTAL_TIPO_MOTONETA_DIFERENCA_MES            0.430809
        TOTAL_TIPO_ÔNIBUS                           -0.311332
        TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES              0.475731
        TOTAL_TIPO_REBOQUE                          -0.329030
        TOTAL_TIPO_REBOQUE_DIFERENCA_MES             0.183569
        TOTAL_TIPO_SEMI_REBOQUE                     -0.339291
        TOTAL_TIPO_SIDE_CAR                          0.285496
        TOTAL_TIPO_OUTROS                           -0.349350
        TOTAL_TIPO_TRATOR_RODAS                     -0.399525
        TOTAL_TIPO_TRICICLO                         -0.365400
        TOTAL_TIPO_UTILITÁRIO                       -0.306616
        TOTAL_TIPO_UTILITÁRIO_DIFERENCA_MES          0.110976
        Name: TOTAL_ACIDENTES_MES, dtype: float64
```

Dessa maneira, foram descartadas as *features* com **coeficiente correlação > 0.38 ou < -0.40**. As *features* com correlação dentro do especificado estão listadas abaixo:

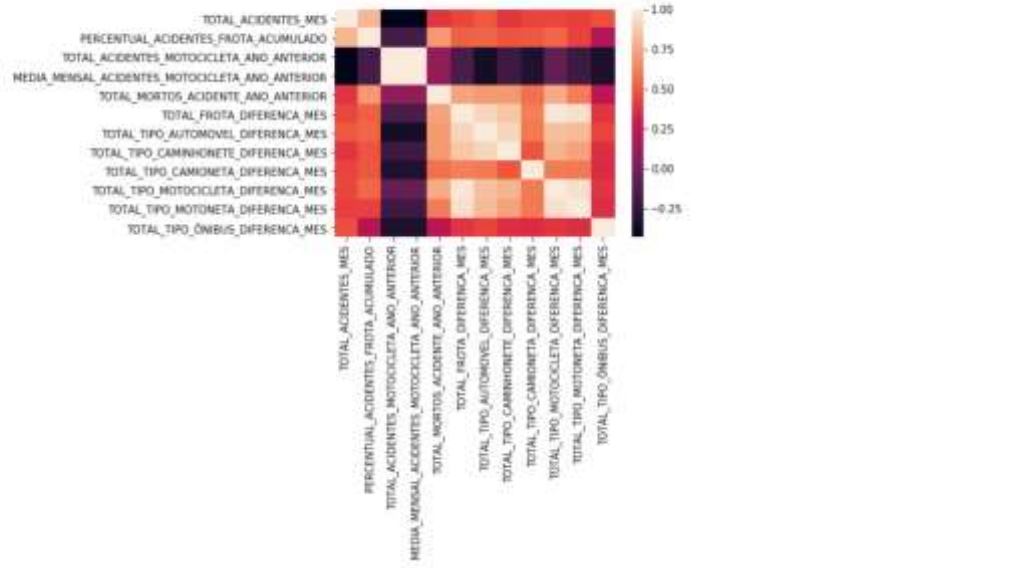
```
In [5]: ## Mostra apenas as features com coeficiente correlacao > 0.38 ou < -0.40
        data_corr.loc[(data_corr['TOTAL_ACIDENTES_MES'] > 0.38) | (data_corr['TOTAL_ACIDENTES_MES'] < -0,40)]['TOTAL_ACIDENTES_MES']

Out[5]: TOTAL_ACIDENTES_MES                          1.000000
        PERCENTUAL_ACIDENTES_FROTA_ACUMULADO         0.806794
        TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR    -0.428317
        MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR -0.425051
        TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR           0.389029
        TOTAL_FROTA_DIFERENCA_MES                    0.448403
        TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES           0.508543
        TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES         0.395721
        TOTAL_TIPO_CAMIONETA_DIFERENCA_MES           0.444738
        TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES         0.448273
        TOTAL_TIPO_MOTONETA_DIFERENCA_MES            0.430809
        TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES              0.475731
        Name: TOTAL_ACIDENTES_MES, dtype: float64
```

Ao final da *feature engineering*, apenas 11 *features* foram consideradas para as fases seguintes de treinamento e teste do modelo preditivo. O *dataset* final ficou com a seguinte estrutura:

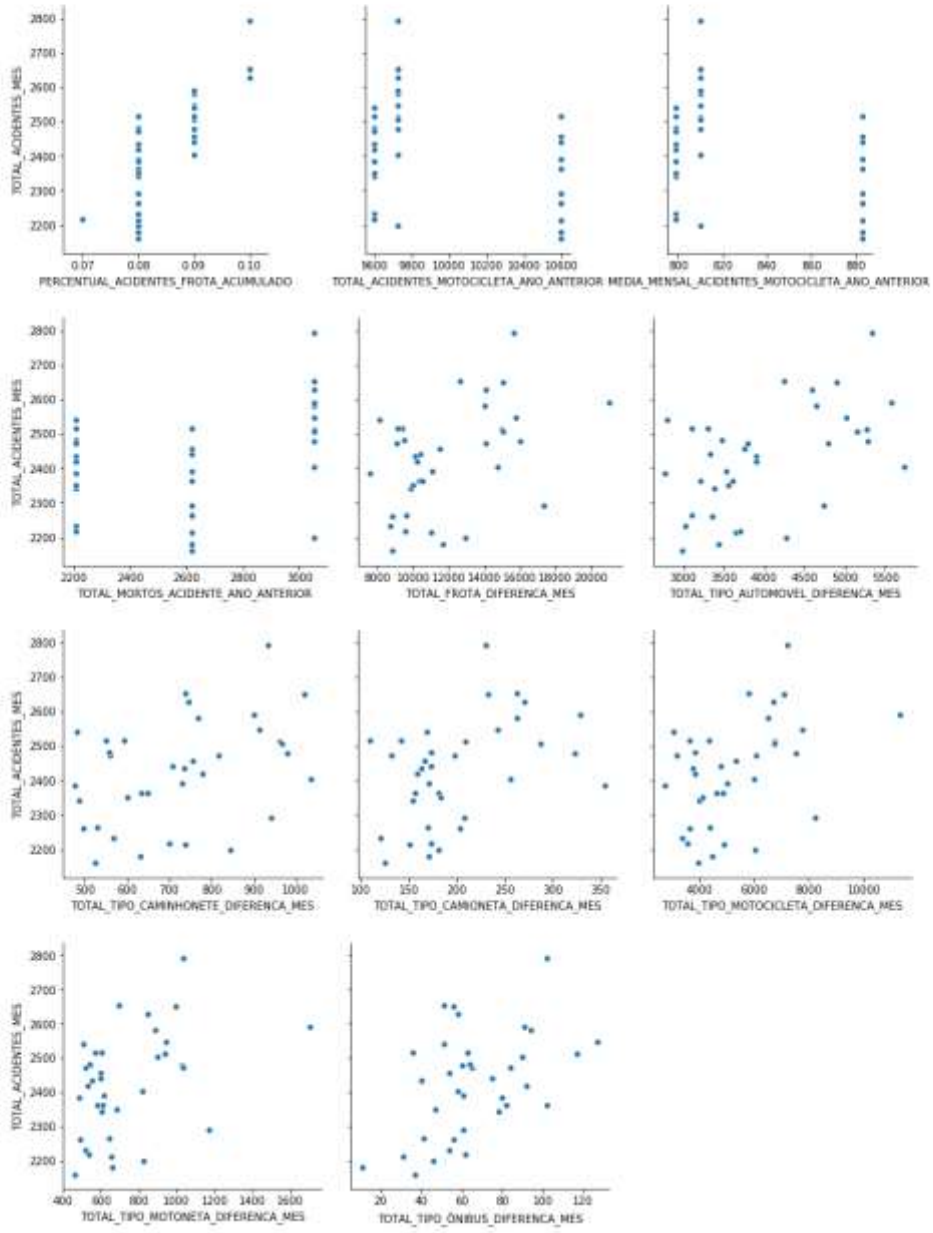
Dataset:	Acidentalidade_DETRAN_CE_2015-2017.csv
Features:	**MES (ANO/MÊS de referência)**
	TOTAL_ACIDENTES_MES (variável Y – saída que se deseja prever)
	PERCENTUAL_ACIDENTES_FROTA_ACUMULADO
	TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR
	MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR
	TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR
	TOTAL_FROTA_DIFERENCA_MES
	TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES
	TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES
	TOTAL_TIPO_CAMIONETA_DIFERENCA_MES
	TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES
	TOTAL_TIPO_MOTONETA_DIFERENCA_MES
	TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES

Vale ressaltar que apesar dessas *features* terem as melhores correlações entre todas, ainda assim são correlações muito baixas, o que poderá prejudicar o modelo de regressão. Abaixo seguem o mapa de calor das correlações entre as variáveis do *dataset* final (as cores representam o coeficiente de correlação), bem como os gráficos de dispersão de cada variável x em relação a y.

```
In [10]: data_corr = data.corr()
         sns.heatmap(data_corr)

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4aae93c7f0>
```



```
In [6]: sns.pairplot(data, height=4, x_vars=['PERCENTUAL_ACIDENTES_FROTA_ACUMULADO',
                                             'TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR',
                                             'MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR'], y_vars=['TOTAL_ACIDENTES_MES'])

        sns.pairplot(data, height=4, x_vars=['TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR',
                                             'TOTAL_FROTA_DIFERENCA_MES',
                                             'TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES'], y_vars=['TOTAL_ACIDENTES_MES'])

        sns.pairplot(data, height=4, x_vars=['TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES',
                                             'TOTAL_TIPO_CAMIONETA_DIFERENCA_MES',
                                             'TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES'], y_vars=['TOTAL_ACIDENTES_MES'])

        sns.pairplot(data, height=4, x_vars=['TOTAL_TIPO_MOTONETA_DIFERENCA_MES',
                                             'TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES'], y_vars=['TOTAL_ACIDENTES_MES'])

Out[6]: <seaborn.axisgrid.PairGrid at 0x7f7f2711bba8>
```



A etapa de *Feature Engineering* teve que ser reexecutada mais 2 vezes. A primeira vez, para tentar encontrar novas *features* que tivessem um melhor coeficiente de correlação com a variável y, dando origem às *features* derivadas:

- TOTAL_FROTA_DIFERENCA_MES
- TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES
- TOTAL_TIPO_CAMINHAO_DIFERENCA_MES
- TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES

- TOTAL_TIPO_CAMIONETA_DIFERENCA_MES
- TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES
- TOTAL_TIPO_MOTONETA_DIFERENCA_MES
- TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES
- TOTAL_TIPO_REBOQUE_DIFERENCA_MES
- TOTAL_TIPO_UTILITÁRIO_DIFERENCA_MES

A segunda vez, para adicionar mais dados ao *dataset*, como uma tentativa de melhorar o desempenho do modelo de dados, visto que aparentemente quanto mais dados se usava no treino, melhor era o resultado da predição em cima dos dados de teste. Foram adicionados dados de 2013/02 até 2014/12 (23 linhas).

Grid Search & Cross Validation

A partir dos dados de treinamento é possível calcular um peso para cada feature, e assim gerar modelos que podem ser utilizados para a predição de uma varável y. O algoritmo que foi usado nesse experimento é ***Random Forest Regression (RFR)*** da biblioteca Sklearn.

O modelo de floresta aleatório é um tipo de modelo aditivo que faz previsões combinando decisões de uma sequência de modelos de base. Mais formalmente, podemos escrever essa classe de modelos como:

$g(x) = f0(x) + f1(x) + f2(x) + . . .$

Onde o modelo final g é a soma de modelos básicos simples f_i. Nele, cada classificador base é uma árvore de decisão simples. Essa técnica de usar vários modelos para obter melhor desempenho preditivo é chamada de agrupamento de modelos. Em florestas aleatórias, todos os modelos base são construídos de forma independente usando uma sub amostra diferente dos dados.

Hiperparâmetros de ajuste

Define-se como os parâmetros de treinamento, que permitem controlar a qualidade do modelo resultante. Embora cada modelo tenha sua configuração padrão, é possível melhorar o seu desempenho preditivo alterando os valores de hiperparâmetros.

Os hiperparâmetros que foram considerados para o algoritmo RFR foram obtidos através do algoritmo GridSearchCV da biblioteca Sklearn. Com ele foi possível realizar um ***Grid Search*** com ***Cross Validation (CV)***, de maneira a permitir encontrar a melhor combinação de hiperparâmetros para o problema que se deseja resolver com base nos dados que se tem.

Os hiperparâmetros do algoritmo ***Random Forest Regression*** que foram considerados são:

1) **n_estimators**: Trata-se do número de árvores na floresta. Esse parâmetro influencia na velocidade (e / ou uso de memória), no entanto quanto mais árvores, maior a precisão. Dessa maneira, deve-se aplicar o ***CV*** para tentar encontra o valor que melhor se ajusta ao *dataset*.

2) **min_samples_split**: Trata-se do número mínimo de amostras necessárias para dividir um nó interno. Esse parâmetro é usado para controlar o *over-fitting*. Valores muito altos impedem que um modelo aprenda relações que podem ser altamente específicas para a amostra particular selecionada em uma árvore. Valores muito altos podem levar a *under-fitting*, portanto, esse parâmetro deve ser ajustado usando o ***CV***.

3) **max_features**: É o número de *features* a serem consideradas em cada divisão. Como regra geral, a raiz quadrada do número total de *features* funciona muito bem, mas devemos verificar até 30-40% do número total

de *features*. Valores mais altos podem levar ao *over-fitting*, dependendo do *dataset*.

4) **min_samples_leaf**: O número mínimo de amostras necessárias para estar em um nó da folha. Um ponto de divisão em qualquer profundidade só será considerado se deixar pelo menos *min_samples_leaf* amostras de treinamento em cada uma das ramificações esquerda e direita. Isso pode ter o efeito de suavizar o modelo, especialmente na regressão.

O modelo de floresta aleatório é muito bom para manipular dados tabulares com *features* numéricas ou *features* categóricas com menos de centenas de categorias. Ao contrário dos modelos lineares, as florestas aleatórias são capazes de capturar a interação não linear entre as *features* e variável y.

Outro ponto a ser observado é que os modelos baseados em árvore não são projetados para funcionar com *features* muito esparsas. Ao lidar com dados de entrada esparsos faz-se necessário pré-processar as *features* esparsas para gerar estatísticas numéricas ou alternar para um modelo linear. Como o *dataset* utilizado no experimento se tem essa característica, foi feita uma normalização dos dados antes de realizar o treinamento.

Essa etapa foi reexecutada várias vezes durante o ajuste do conjunto de valores de hiperparâmetros que serviram de entrada para o GridSearchCV. Esses ajustes foram realizados tanto no *dataset* original, quanto depois que ele foi aumentado (23 linhas).

Após essa reexecução os melhores hiperparâmetros encontrados foram:

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

In [3]:
```
## Importa o csv com os dados de acidentes
#data = pd.read_csv('Acidentalidade_DETRAN_CE_2015-2017.csv')
data = pd.read_csv('Acidentalidade_DETRAN_CE_2013-2017.csv')

## Gera o dataframe apenas com as features de maior relevancia
data = data[[ 'MES',
              'TOTAL_ACIDENTES_MES',
              'PERCENTUAL_ACIDENTES_FROTA_ACUMULADO',
              'TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR',
              'MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR',
              'TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR',
              'TOTAL_FROTA_DIFERENCA_MES',
              'TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES',
              'TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES',
              'TOTAL_TIPO_CAMIONETA_DIFERENCA_MES',
              'TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES',
              'TOTAL_TIPO_MOTONETA_DIFERENCA_MES',
              'TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES'
            ]]

data = data.drop('MES',1)

X = data.drop('TOTAL_ACIDENTES_MES',1)
y = data['TOTAL_ACIDENTES_MES']

X.head() ## mostra apenas algumas linhas
```

Out[3]:

	PERCENTUAL_ACIDENTES_FROTA_ACUMULADO	TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR	MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTE
0	0.09	9429.0	
1	0.10	9429.0	
2	0.10	9429.0	
3	0.11	9429.0	
4	0.10	9429.0	

In [4]:
```
# Separa os dados de treino e teste (70% treino e 30% teste)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=000)
```

In [5]:
```
# Normaliza os dados entre -1 and 1
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [6]:
```
rfr = RandomForestRegressor()
```

In [10]:
```
############ Cross Validation & Grid Search - Busca da melhor combinação de hiperparâmetros ##################

param_grid = {'n_estimators': [200, 300, 400],  # primeiro engano: o tamanho da floresta deve ser o maximo possivel
              'max_features': ['auto', 'log2', 'sqrt', 2], # auto = num_features
              'min_samples_split': [2, 0.01, 0.05, 0.1],  # 2 samples, 1% of samples, 5% of samples and 10%
              'min_samples_leaf': [5, 0.01, 0.05, 0.1],
              'min_impurity_split': [None],
              'oob_score': [True],
              'n_jobs': [-1],
              'random_state': [1]
             }

#Lista de scorring (parâmetro que controla quais métricas serão aplicadas aos estimadores avaliados)
scoring = {'R2':'r2', 'MAE':'mean_absolute_error', 'MSE':'mean_squared_error'}
gs = GridSearchCV(rfr, param_grid, verbose=1, cv=10, n_jobs=-1, scoring=scoring, iid=False, refit='MAE')
gs = gs.fit(X, y) # Perform grid search for each algorithm with 10 folds and run fit with all sets of parameters

print(gs.best_params_) # Show the best parameters from the executed algorithm
```

```
Fitting 10 folds for each of 192 candidates, totalling 1920 fits
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   25.3s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  3.7min
[Parallel(n_jobs=-1)]: Done 792 tasks      | elapsed:  6.4min
[Parallel(n_jobs=-1)]: Done 1242 tasks      | elapsed:  9.3min
[Parallel(n_jobs=-1)]: Done 1792 tasks      | elapsed: 12.3min
[Parallel(n_jobs=-1)]: Done 1920 out of 1920 | elapsed: 13.0min finished
{'max_features': 'auto', 'min_impurity_split': None, 'min_samples_leaf': 0.01, 'min_samples_split': 0.1, 'n_estimators': 400,
'n_jobs': -1, 'oob_score': True, 'random_state': 1}
```

Training & Test

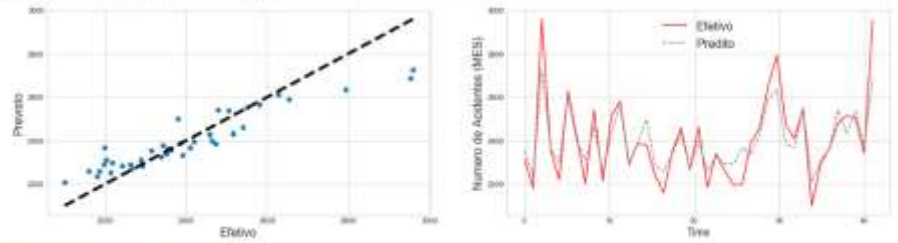
Após os hiperparâmetros serem encontrados na etapa anterior, eles foram aplicados no treinamento e teste do modelo de regressão. Os dados do dataset foram divididos em 70% para treino e 30% para teste.

```
In [11]: rfr = RandomForestRegressor(max_features= 'auto', min_impurity_split= None, min_samples_leaf= 0.01, min_samples_split= 0.1,
                                     n_estimators= 400, n_jobs= -1, oob_score= True, random_state= 1)

         rfr.fit(X_train, y_train.values.reshape(-1,))

Out[11]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=0.01, min_samples_split=0.1,
                    min_weight_fraction_leaf=0.0, n_estimators=400, n_jobs=-1,
                    oob_score=True, random_state=1, verbose=0, warm_start=False)
```

Com o modelo treinado, foram aplicadas duas predições: a primeira sobre os dados do próprio treinamento, como forma de validar o treino, e a segunda sobre os dados de teste, como forma medir a capacidade de generalização do modelo.

<u>Predição sobre os dados de treinamento:</u>

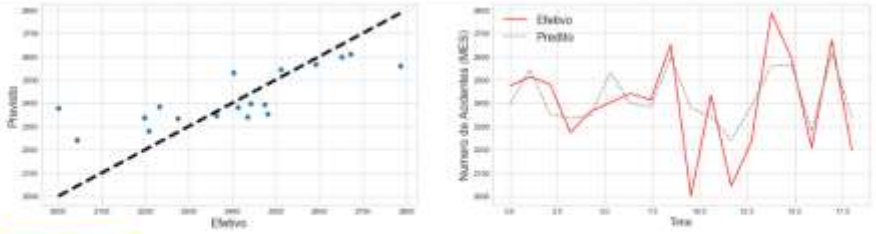
```
[13]: # Com o modelo criado, faz a predição em cima dos dados de TREINO (validação)
      y_pred_train = rfr.predict(X_train)

      # Gera as métricas de validação do modelo com base em y_train
      #MSE  = metrics.mean_squared_error(y_train, y_pred_train)
      #RMSE = np.sqrt(MSE)
      MAE  = metrics.mean_absolute_error(y_train, y_pred_train)
      R2   = metrics.r2_score(y_train, y_pred_train)

      # Plota os resultados da predição sobre os dados de treino
      plt.style.use('seaborn-whitegrid')
      fig, ax = plt.subplots(1,2, figsize=(20,5))
      fig.subplots_adjust(hspace=0.2, wspace=0.2)


      ax[0].scatter(y_train, y_pred_train)
      ax[0].plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'k--', lw=4)
      ax[0].set_xlabel('Efetivo',fontsize=16)
      ax[0].set_ylabel('Previsto',fontsize=16)

      xfit = np.linspace(0,X_train.shape[0], X_train.shape[0])
      yfit = y_pred_train
      ax[1].plot(xfit, y_train, '-r')
      ax[1].plot(xfit, yfit, '--b', color='gray')
      ax[1].set_xlabel('Time',fontsize=14)
      ax[1].set_ylabel('Numero de Acidentes (MES)',fontsize=16)
      ax[1].legend(('Efetivo','Predito'),fontsize=16)

      plt.show()
      plt.gcf().clear()

      print('R2 score: %f' % (R2))
      print('Mean Absolute Error: %f' % (MAE))
      #print('Mean Square Error: %f' % (MSE))
      #print('Root Mean Square Error: %f' % (RMSE))
```




```
R2 score: 0.983822
Mean Absolute Error: 63.414374
```

<u>Predição sobre os dados de teste:</u>

```
In [14]: # Com o modelo criado, faz a predição em cima dos dados de TESTE
y_pred_test = rfr.predict(X_test)

# Gera as métricas de validação do modelo com base em y_test
MSE  = metrics.mean_squared_error(y_test, y_pred_test)
#RMSE = np.sqrt(MSE)
MAE  = metrics.mean_absolute_error(y_test, y_pred_test)
R2   = metrics.r2_score(y_test, y_pred_test)

# Plota os resultados da predição sobre os dados de teste
plt.style.use('seaborn-whitegrid')
fig, ax = plt.subplots(1,2, figsize=(20,5))
fig.subplots_adjust(hspace=0.2, wspace=0.2)


ax[0].scatter(y_test, y_pred_test)
ax[0].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
ax[0].set_xlabel('Efetivo',fontsize=16)
ax[0].set_ylabel('Previsto',fontsize=16)

xfit = np.linspace(0,X_test.shape[0], X_test.shape[0])
yfit = y_pred_test
ax[1].plot(xfit, y_test, '-r')
ax[1].plot(xfit, yfit, '--b', color='gray')
ax[1].set_xlabel('Time',fontsize=14)
ax[1].set_ylabel('Numero de Acidentes (MES)',fontsize=16)
ax[1].legend(('Efetivo','Predito'),fontsize=16)

plt.show()
plt.gcf().clear()

print('R2 score: %f' % (R2))
print('Mean Absolute Error: %f' % (MAE))
#print('Mean Square Error: %f' % (MSE))
#print('Root Mean Square Error: %f" % (RMSE))
```




```
R2 score: 0.548476
Mean Absolute Error: 186.451282
```

Essa etapa foi reexecutada inúmeras vezes, devido ao baixo desempenho do modelo na predição sobre os dados de teste. Foram simuladas diversas divisões do *dataset* em treino em teste, de maneira que se observou que o aumento do tamanho dos dados de treinamento melhorava o resultado dos testes. No entanto, dado ao tamanho do *dataset* (apenas 3 anos – 36 linhas), existia um limite para diminuição dos dados de teste. Dessa maneira, optou-se por coletar mais dados. Foram adicionados ao *dataset* dados de 2013/02 até 2014/12 (23 linhas).

Apesar do aumento ter melhorado os resultados da predição sobre os dados de teste, ele não foi suficiente para resolver o problema e os resultados abaixo do aceitável.

Dessa maneira, vale lembrar que já fase de *Feature Engineering* foi constatada a baixa correlação das *features* com a variável de saída y – mesmo aumentando o *dataset* – o que poderia prejudicar o modelo de regressão.

```
In [5]: ## Mostra apenas as features com coeficiente correlacao > 0.38 ou < -0.40
data_corr.loc[(data_corr['TOTAL_ACIDENTES_MES'] > 0.38) | (data_corr['TOTAL_ACIDENTES_MES'] < -0.40)]['TOTAL_ACIDENTES_MES']

Out[5]: TOTAL_ACIDENTES_MES                          1.000000
        PERCENTUAL_ACIDENTES_FROTA_ACUMULADO         0.806794
        TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR    -0.428317
        MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR -0.425051
        TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR           0.389029
        TOTAL_FROTA_DIFERENCA_MES                    0.448403
        TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES           0.508543
        TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES         0.395721
        TOTAL_TIPO_CAMIONETA_DIFERENCA_MES           0.444738
        TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES         0.448273
        TOTAL_TIPO_MOTONETA_DIFERENCA_MES            0.430889
        TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES              0.475731
        Name: TOTAL_ACIDENTES_MES, dtype: float64
```

O *feature_importances_* é uma estimativa para qual fração da classificação de amostras de entrada uma *feature* contribui para o modelo.

Por exemplo, 1.0 significaria que você tem uma *feature* que classifica sozinha todas as amostras, 0.0 indicaria uma *feature* que pode adicionar nenhum valor (adicional) para classificação.

Se listarmos o coeficiente de importância das *features* para o modelo, veremos que a maioria tem relevância baixa.

```
In [12]: # lista as features de maior importancia
         list(zip(X.columns, rfr.feature_importances_))
```

```
Out[12]: [('PERCENTUAL_ACIDENTES_FROTA_ACUMULADO', 0.3498941110201902),
          ('TOTAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR', 0.004678536003119646),
          ('MEDIA_MENSAL_ACIDENTES_MOTOCICLETA_ANO_ANTERIOR', 0.003521260199042854),
          ('TOTAL_MORTOS_ACIDENTE_ANO_ANTERIOR', 0.2216085599615172),
          ('TOTAL_FROTA_DIFERENCA_MES', 0.05338080075761846),
          ('TOTAL_TIPO_AUTOMOVEL_DIFERENCA_MES', 0.06543684240642708),
          ('TOTAL_TIPO_CAMINHONETE_DIFERENCA_MES', 0.059174470574515976),
          ('TOTAL_TIPO_CAMIONETA_DIFERENCA_MES', 0.06333178631629906),
          ('TOTAL_TIPO_MOTOCICLETA_DIFERENCA_MES', 0.034217604526612905),
          ('TOTAL_TIPO_MOTONETA_DIFERENCA_MES', 0.06340178938258188),
          ('TOTAL_TIPO_ÔNIBUS_DIFERENCA_MES', 0.08135423915207447)]
```

Conclusão

Acredito que os fatores mais relevantes para o resultado negativo do experimento, se deu principalmente pelo fato do *dataset* ser muito esparso e pequeno (granularidade mensal), além da baixa correlação das *features* com variável y. Outra hipótese é que as *features* que teriam mais relevância para o modelo seriam as que tem relação com o condutor (Aspectos do Condutor), que infelizmente não puderam ser coletadas, dado que não foram encontrados dados abertos na web nesse aspecto.

Sobre o experimento em si, foi possível constatar o quão é importante a mineração dos dados para o aprendizado de máquina, dado que ela intensifica a capacidade preditiva dos algoritmos, por meio da seleção das melhores *features* para o contexto do problema.

Referências

https://sidra.ibge.gov.br/tabela/1419

https://sidra.ibge.gov.br/tabela/1100

http://idg.receita.fazenda.gov.br/dados/receitadata/arrecadacao/arrecadacao-por-estado

https://www.denatran.gov.br/estatistica/237-frota-veiculos

http://portal.detran.ce.gov.br/index.php/estatisticas/estatisticas-de-acidentes

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

https://turi.com/learn/userguide/supervised-learning/random_forest_regression.html

http://www.montefiore.ulg.ac.be/~glouppe/pdf/phd-thesis.pdf