Manutenção de Viaturas

ENAP/Bootcamp Machine Learning/2021

Polícia Rodoviária Federal
Anderson Martins Gomes

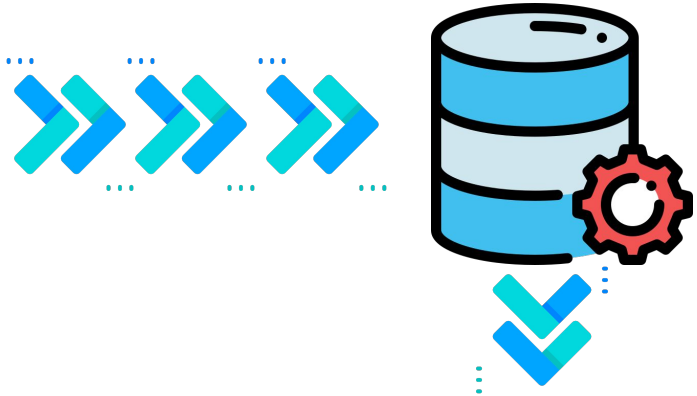
Problema de Negócio

- +70.000 km de rodovias federais
- +-13.000 servidores
- +700 pontos fixos de fiscalização
- +4500 viaturas em circulação
- 27 gestões regionais

Solução

Machine Learning

Dados



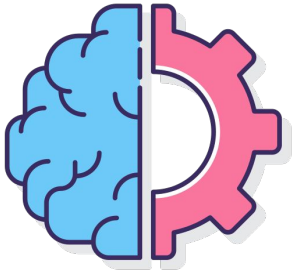
Situação Viatura	Grave?
Tudo ok!	NÃO
Freio falhando	**SIM**
Porta amassada	NÃO
Suspensão com barulho esquisito	**SIM**

Treinamento e Avaliação

Situação Viatura	Grave?
Tudo ok!	NÃO
Freio falhando	SIM
Porta amassada	NÃO
Suspensão com barulho esquisito	SIM

+3000 registros
Máx 2095 char/registro

+4000 features
15 algoritmos
Hiper-parâmetros



TF-IDF
Métodos Ensemble
Feature Engineering
Cross Validation
BayesSearch
Algoritmo Genético
AutoML

	train_order	algorithm	
0	437	StackingClassifier(estimators=[('RandomForestC...	(42
1	221	StackingClassifier(estimators=[('RandomForestC...	(42
2	314	StackingClassifier(estimators=[('RandomForestC...	(42
3	318	StackingClassifier(estimators=[('RandomForestC...	(42
4	341	StackingClassifier(estimators=[('RandomForestC...	(42
...	...	...	
696	456	MultinomialNB()	(42
697	246	MultinomialNB()	(42
698	231	MultinomialNB()	(42
699	611	MultinomialNB()	(42
700	0	KNeighborsClassifier()	(42

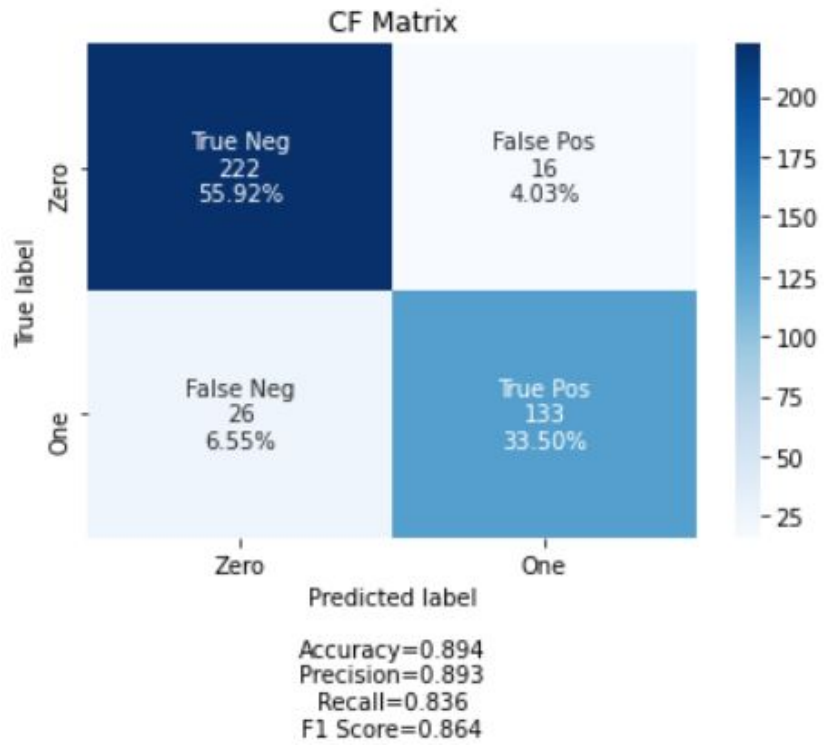
701 rows × 10 columns

Treinamento e Avaliação

```
Original dataset dimensions: (1984, 4379)
Dataset dimensions after drop NaN values: (1984, 4379)
ML problem type: Classification
   Applied metrics: ['roc_auc', 'f1', 'accuracy']
Normalizing the variables...
Splitting dataset...
   X_train dimensions: (1587, 4378)
Features engineering - Testing correlation with Y...
   Features engineering - Features reduction after correlation test with Y: 95.45% (199 remained)
Features engineering - Testing redudance between features...
   Features engineering - Features reduction after redudance test: 95.50% (197 remained)
Selected algorithms: ['KNeighborsClassifier', 'SVC', 'GaussianProcessClassifier', 'DecisionTreeClassifier', 'RandomForestClassifier', 'AdaBoostClassifi
Nº of training possible combinations: 6.026017665971686e+60 (4.017345110647791e+59 features combinations, 15 algorithms)
   *Model trained: roc_auc = 0.84602 | 197 features | KNeighborsClassifier | ('4275', '4274', '4255', '4253', '4251', '4247', '4223', '4218', '4195',
```

	algorithm	features	n_features	train_time	mem_max	roc_auc	f1	accuracy	confusion_matrix
0	VotingClassifier(estimators=[('e1', RandomFore...	('4275', '4274', '4255', '4253', '4251', '4247...	183	100.752640	2673.277344	0.953570	0.863636	0.894207	[[222 16]\n [26 133]]
1	VotingClassifier(estimators=[('e1', RandomFore...	('4274', '4255', '4253', '4251', '4247', '4223...	180	94.176428	2682.113281	0.952619	0.858065	0.889169	[[220 18]\n [26 133]]
2	StackingClassifier(estimators=[('e1', RandomFo...	('4275', '4274', '4255', '4253', '4223', '4218...	169	695.589180	2602.246094	0.947069	0.857143	0.886650	[[217 21]\n [24 135]]
3	VotingClassifier(estimators=[('e1', RandomFore...	('4275', '4274', '4255', '4253', '4251', '4247...	180	108.023518	2606.433594	0.949818	0.849673	0.884131	[[221 17]\n [29 130]]
4	VotingClassifier(estimators=[('e1', RandomFore...	('4275', '4274', '4255', '4253', '4251', '4247...	195	124.971950	2364.562500	0.953702	0.852564	0.884131	[[218 20]\n [26 133]]
5	VotingClassifier(estimators=[('e1', RandomFore...	('4274', '4255', '4251', '4247', '4223', '4218...	183	115.882148	2603.609375	0.943423	0.852564	0.884131	[[218 20]\n [26 133]]
6	VotingClassifier(estimators=[('e1', RandomFore...	('4275', '4274', '4255', '4253', '4251', '4247...	175	111.037226	2675.500000	0.951641	0.850649	0.884131	[[220 18]\n [28 131]]
7	RandomForestClassifier(n_jobs=-1)	('4275', '4274', '4255', '4253', '4251', '4247...	197	118.888302	1970.925781	0.944982	0.850649	0.884131	[[220 18]\n [28 131]]
8	StackingClassifier(estimators=[('e1', RandomFo...	('4275', '4274', '4255', '4253', '4251', '4223...	182	437.815891	2761.882812	0.947783	0.854430	0.884131	[[216 22]\n [24 135]]

Interpretação do Modelo



Resultados

- Melhoria de **89,4%** na eficiência global do processo

- Mais segurança / Menos custo

- Biblioteca reutilizável (automl), pública e disponível para a comunidade para resolução de outros problemas de Machine Learning (https://github.com/andersonmgomes/automl)

Dúvidas?

Automl

```
1  from autoML import AutoML
2  from multiprocessing import Pool
3
4  pool = Pool(processes=10)
5  automl = AutoML(dsViaturas, 'y', min_x_y_correlation_rate=0.05, pool=pool, ds_name='viaturas_ngen10_f1', ngen=10)
6  automl.getResults()
```

✓ 179m 5.6s

```
def __init__(self, ds_source, y_colname = 'y'
             , algorithms = ALGORITHMS
             , unique_categoric_limit = 10
             , min_x_y_correlation_rate = 0.01
             , n_features_threshold = 1
             , pool = None
             , ds_name = None
             , ngen = 10) -> None:
```

Automl

```python
def evaluation(individual, automl_obj):
    def float2bigint(float_value):
        if math.isnan(float_value):
            float_value = -1
        return [int(float_value*100000)]

    #print(individual)

    algo = individual[-automl_obj.n_bits_algos:]
    algo = bautil.ba2int(bitarray(algo)) % len(automl_obj.selected_algos)

    algo = automl_obj.selected_algos[algo]

    col_tuple = individual[:len(automl_obj.X_bitmap)-automl_obj.n_bits_algos]
    col_tuple = tuple([automl_obj.X_train.columns[i] for i, c in enumerate(col_tuple) if c == 1])

    if len(col_tuple)==0:
        return float2bigint(-1)

    def is_ensemble(a):
        return isinstance(a, VotingClassifier) or isinstance(a, StackingClassifier)

    if is_ensemble(algo):
        #getting the top 3 best results group by algorithm
        best_estimators = []
        automl_obj.results.sort_values(by=automl_obj.main_metric, ascending=False, inplace=True)
        for row in automl_obj.results.iterrows():
            if len(best_estimators)==3:
                break
            candidate_algo = row[1]['algorithm']
            if ((candidate_algo not in best_estimators)
                and (not is_ensemble(candidate_algo))):
                best_estimators.append(candidate_algo)
        algo.estimators = list(zip(['e'+str(i) for i in range(1,len(best_estimators)+1)],best_estimators))
```

Automl

```
class AutoML:
    ALGORITHMS = {
        #classifiers
        #https://scikit-learn.org/stable/auto_examples/classification/
        KNeighborsClassifier(n_jobs=-1):
            {"n_neighbors": [3,5,7,9,11,13,15,17],
             "p": [2, 3],
             },
        SVC(probability=True):
            {"C": [0.001, 0.01, 0.1, 1, 10, 100, 1000],
             "gamma": ["auto", "scale"],
             "class_weight": ["balanced", None]},
        GaussianProcessClassifier(n_jobs=-1):{
            "copy_X_train": [False],
            "warm_start": [True, False],},
        DecisionTreeClassifier():{
            "criterion": ["gini", "entropy"],
            },
        RandomForestClassifier(n_jobs=-1):{
            "n_estimators": [120,300,500,800,1200],
            "max_depth": [None, 5, 8, 15, 20, 25, 30],
            "min_samples_split": [2, 5, 10, 15, 100],
            "min_samples_leaf": [1, 2, 5, 10],
            "max_features": [None, "sqrt", "log2"],
            },
        #MLPClassifier():{
        #    "activation": ["identity", "logistic", "tanh", "relu"],
        #    },
        AdaBoostClassifier():{
            "algorithm": ["SAMME", "SAMME.R"],
            },
        GaussianNB():{
            "priors": [None],
            },
        QuadraticDiscriminantAnalysis():{
            "priors": [None],
```

```
#tunning parameters
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    opt = BayesSearchCV(estimator=model, search_spaces=automl_obj.algorithms[model],
                        scoring='f1_weighted', n_iter=30, cv=5,
                        verbose=0, n_jobs=-1)
    opt.fit(X_train2, automl_obj.y_train)
model = opt.best_estimator_

metrics_value_list = []
for scor_str in scoring_list:
    metrics_value_list.append(get_scorer(scor_str)(model, X_test2, automl_obj.y_test))

result_list = metrics_value_list

if automl_obj.YisCategorical():
    #confusion matrix
    result_list.append(confusion_matrix(automl_obj.y_test, model.predict(X_test2)))

#model
result_list.append(model)
```