Code Snippet Classification

Big Data Computing Project

Andrea Trianni - Emanuele Mercanti

The Task

Given a **short** snippet of code, predict the programming language.

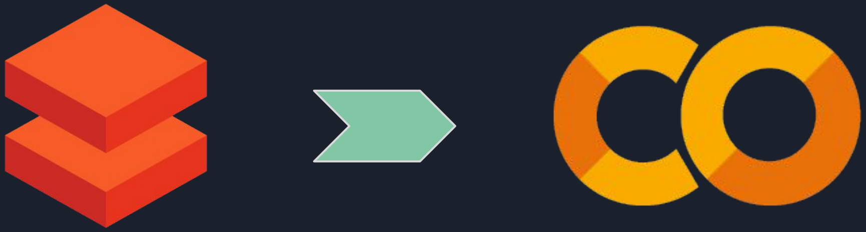
Useful for Visual IDE tool :

- to highlight keywords
- to check syntax error

Useful for Online forums :

- clusters un-tagged questions

The environment



Development

Training stage

The Dataset

GitHub SQL Dataset

Version available: FULL (*60GB*) and LITE (*3GB)*

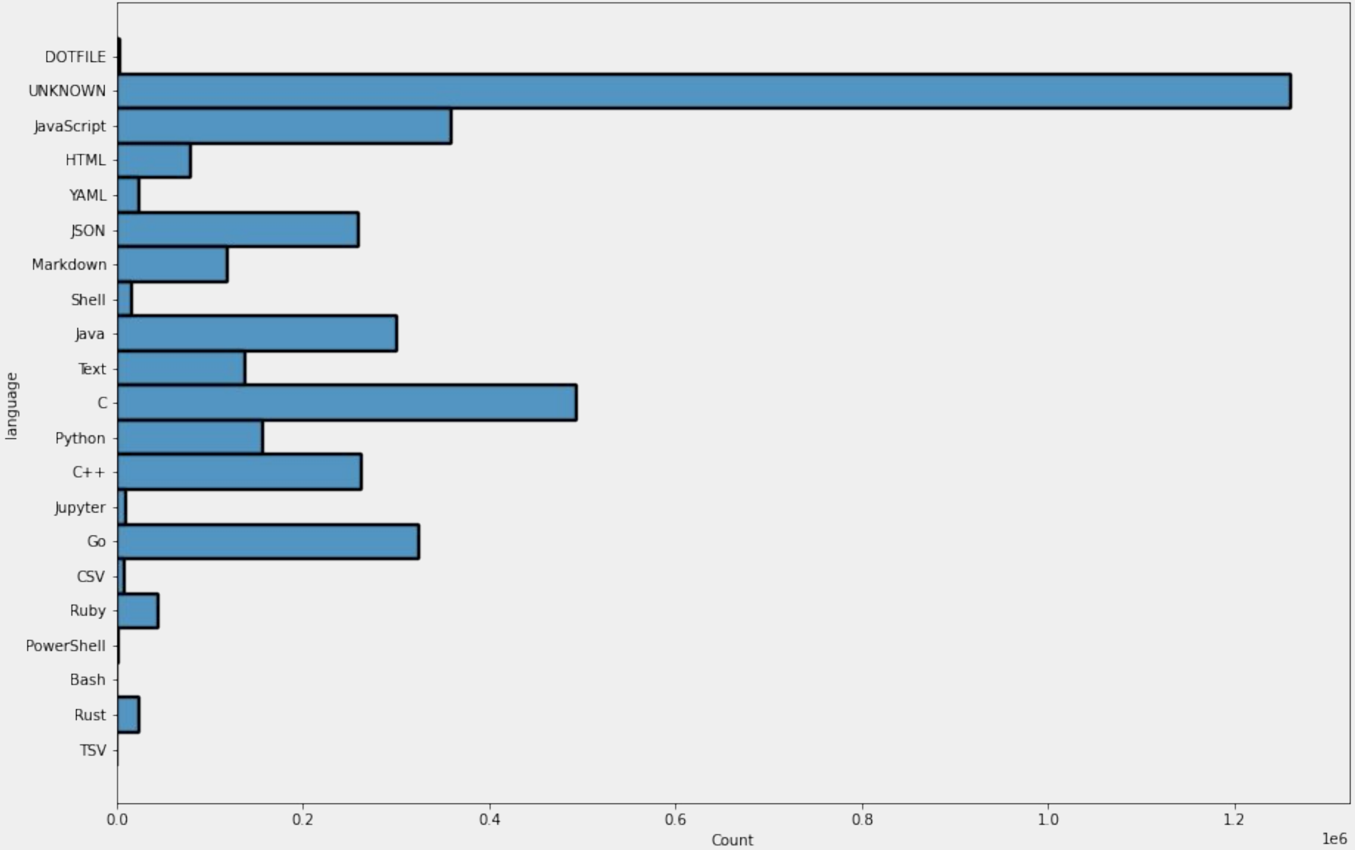
20 Different languages covered + Unknown Class

Length of snippets is **5** rows

We have access also to other Fields, like License Type

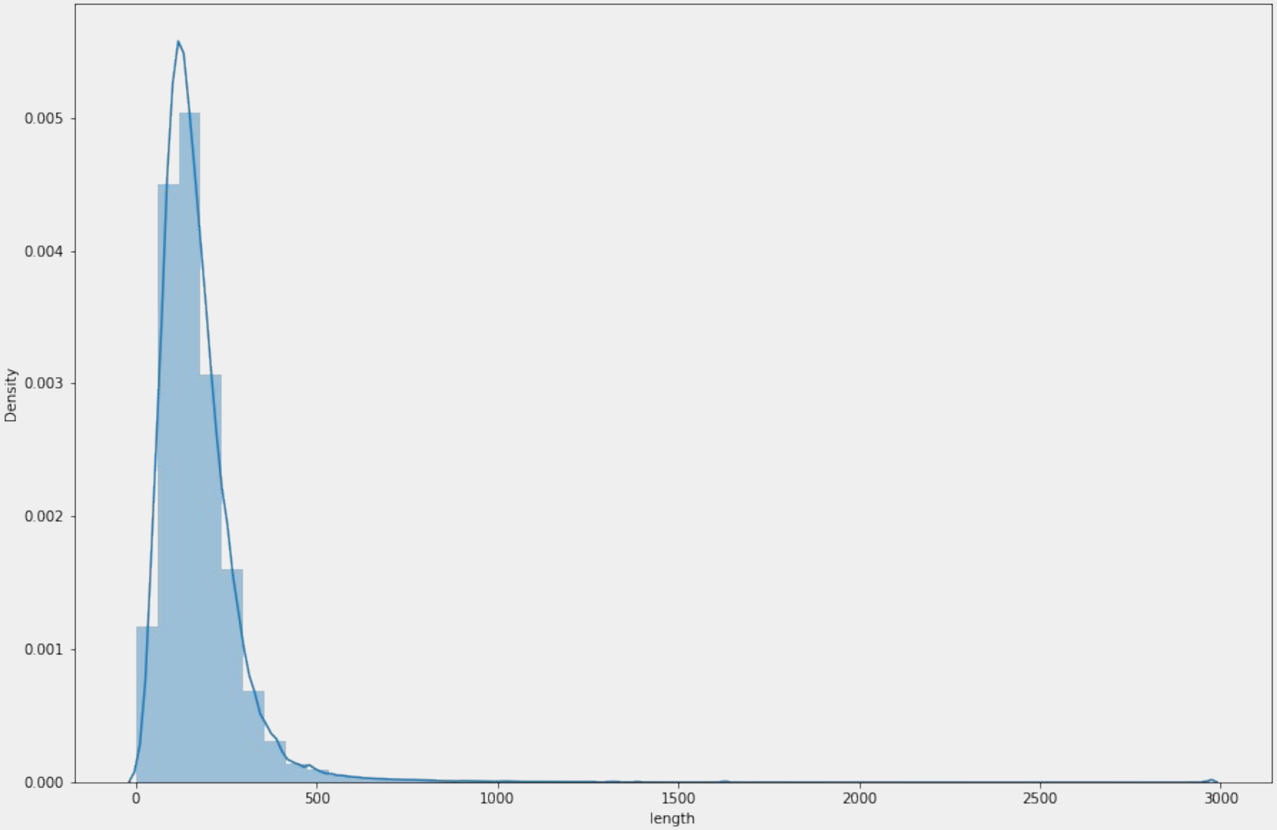
Visualize the data

class sizes



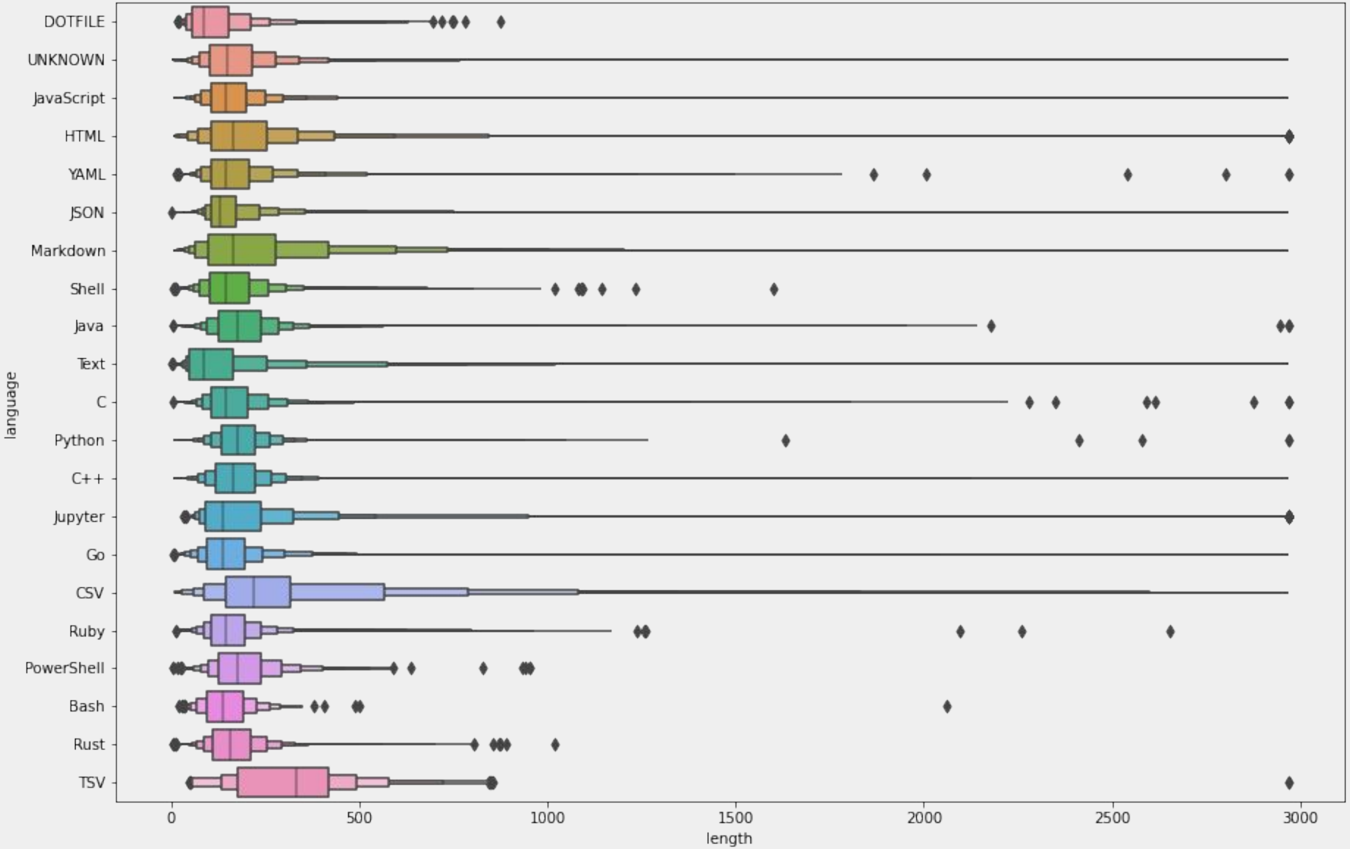
Visualize the data

snippet length distribution



Visualize the data

boxenplot: snippet length distribution by class



Our Approach

The preprocessing pipeline

Cleaning the data

Manual **stratify split** in train, dev and test set

Remove **non-programming** languages (csv, json, ..)
Remove **noisy unknown** class
Merge **Shell** family languages (shell, bash, ..)

Remove **outlier** rows
Check for **missing** data
Calculating class **weights**

Tokenize the snippet

Our first try:

A small Vocabulary manually built from a fixed set of keywords + punctuation

Tokenizing finding substring matches, using the vocabulary.

Strength:

Low dimensionality of vectors helps simple models to achieve high performance

Weakness:

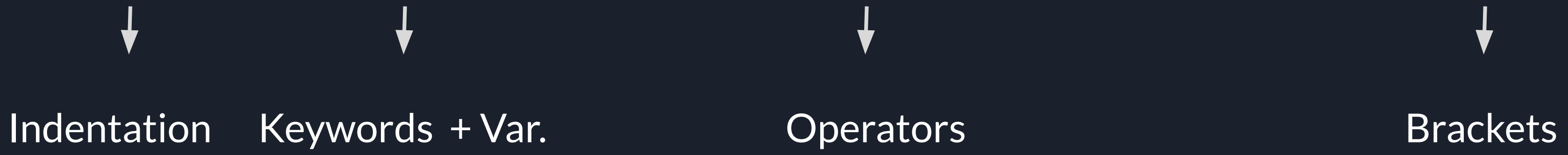
Powerful models not able to reach their best performances

No parallelization possibile with simple implementation

Tokenize the snippet

So, we rely on Spark **RegexTokenizer** :

```
( [\\n\\t] | ([A-Za-z_]+\\b) | [!\#\$%\&\*\+:\-\./<=>\?@\^_\|\~]+ | [ \(\),;\{\}\[\]`\"'] )
```



Indentation

Keywords + Var.

Operators

Brackets

Encoding the snippet

We tried 2 encoding strategy :

✓ BOW ✗ TF-IDF

5k Max Vocabulary Size

Sometimes, less is more.

Our Approach

The training stage

Training the models

We train and evaluate the following models:

Decision tree classifier

Gradient-boosted tree classifier [OVR] (failed)

Random forest

Multinomial logistic regression

Naive Bayes classifier

Linear Support vectors machines [OVR]

Evaluating the models

We obtain the following results:

MODEL	F1 Score (dev)	Accuracy (dev)
Decision tree classifier	0.38	0.45
Gradient-boosted tree	?	?
Random forest	0.55	0.53
Multinomial logistic reg.	**0.83**	**0.83**
Naive bayes	0.74	0.73
Linear SVM	0.82	0.82

Results explanation

Winners

Logistic regression

Linear SVM

Data is linearly separable

Data can be modeled by a multinomial distribution

Losers

Decision tree

Random Forest

Data is very sparse

Data might not be aligned to axis

Benchmark the model

So, we select the best model and we benchmark it on **test** set, using:

Cross Validation + Grid Search

Accuracy + F1 score

Confusion matrix

Learning curve

Results

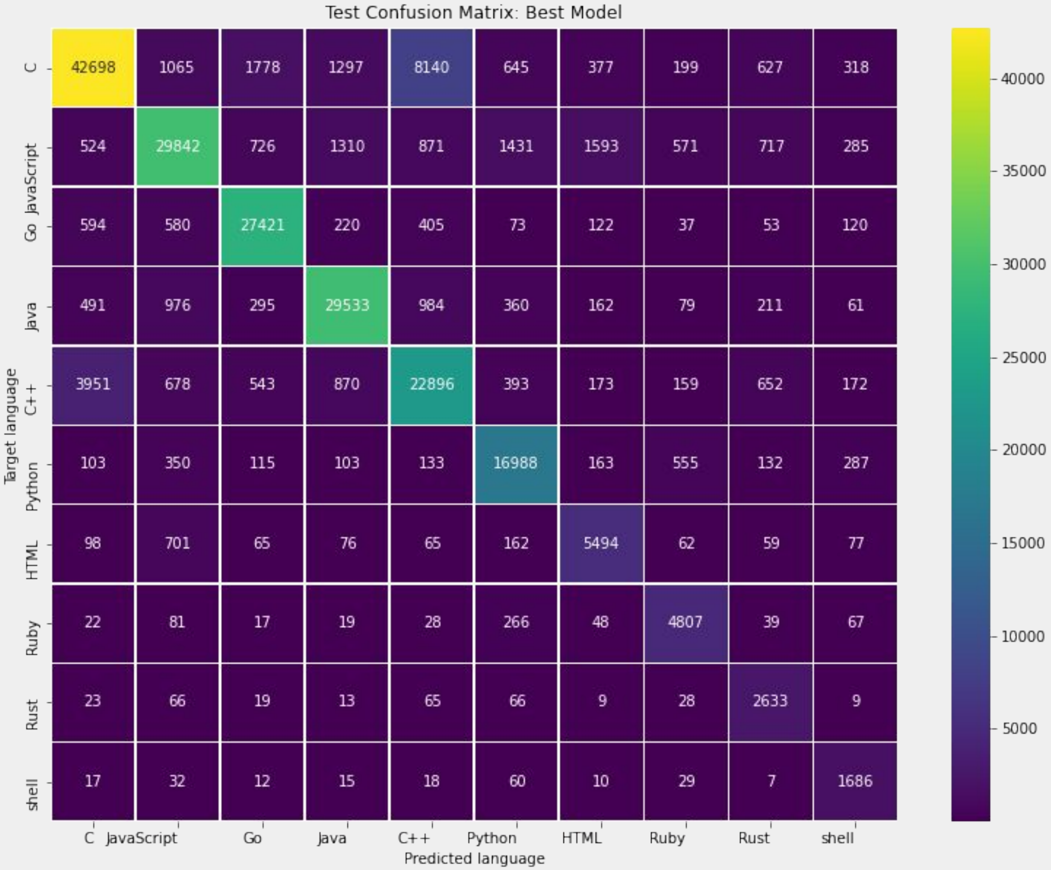
Benchmarks:

(Grid Search) 5 Fold Cross-Validation F1: **82**.5%

Test set results:

F1-Score:	**82.2 %**
Accuracy:	**82 .0%**

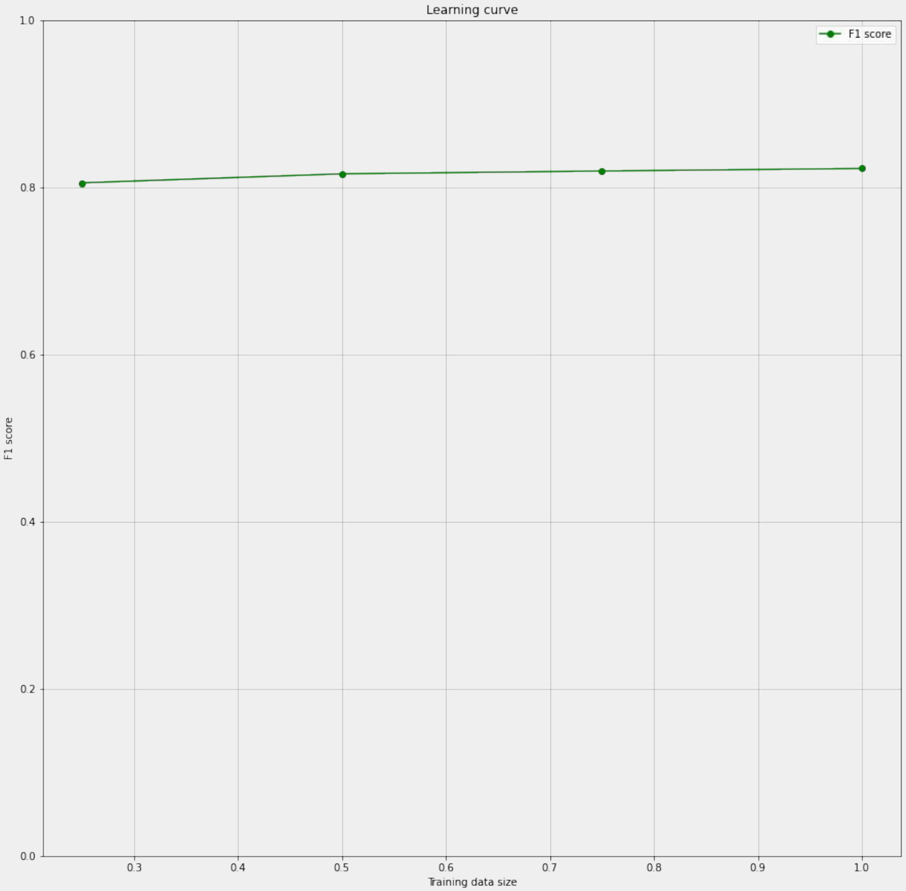
Confusion Matrix



Test Confusion Matrix: Best Model

	C	JavaScript	Go	Java	C++	Python	HTML	Ruby	Rust	shell
C	42698	1065	1778	1297	8140	645	377	199	627	318
JavaScript	524	29842	726	1310	871	1431	1593	571	717	285
Go	594	580	27421	220	405	73	122	37	53	120
Java	491	976	295	29533	984	360	162	79	211	61
C++	3951	678	543	870	22896	393	173	159	652	172
Python	103	350	115	103	133	16988	163	555	132	287
HTML	98	701	65	76	65	162	5494	62	59	77
Ruby	22	81	17	19	28	266	48	4807	39	67
Rust	23	66	19	13	65	66	9	28	2633	9
shell	17	32	12	15	18	60	10	29	7	1686

Learning Curve



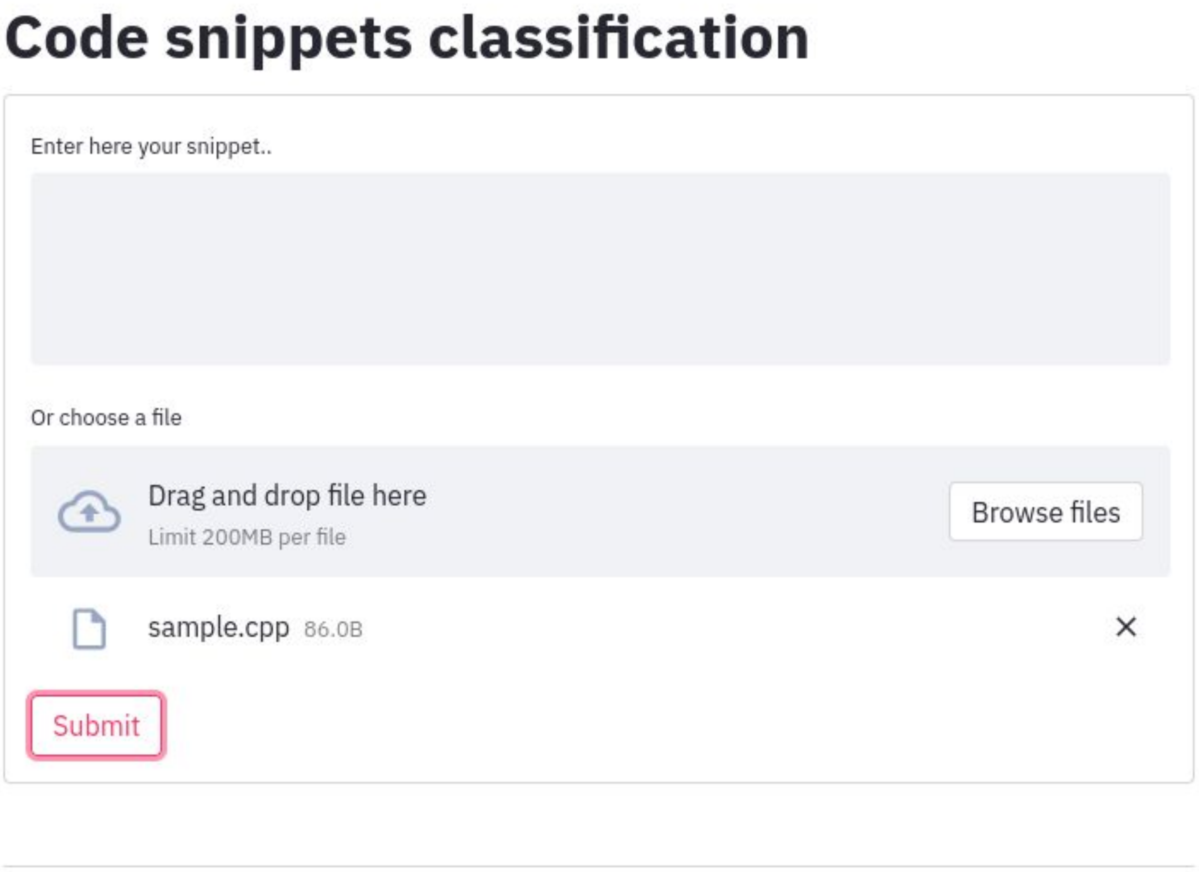
Web Application

How Does it Works?

Instead of leveraging Docker (local) and build a Web Application using Flask REST API..

We decided to use Colab Runtime as a virtual machine, configure Spark with a Master Node and deploy our application using **Streamlit**

Actually, you can try to use it (the link will be provided during the live presentation)



Code snippets classification

Enter here your snippet..

Or choose a file

Drag and drop file here

Limit 200MB per file

sample.cpp 86.0B

Thank you!