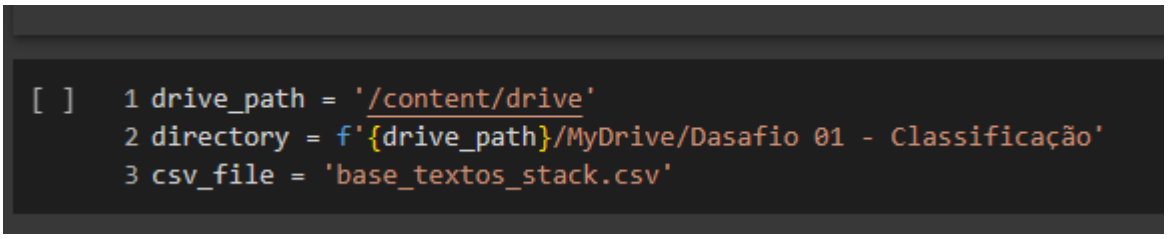
README: Entrega Desafio 1

A entrega do desafio um consiste em três documentos, sendo o *EAD.ipynb, Preprocessamento.ipynb* e *algoritmo_genetico.ipynb.* Os colabs devem ser executados nessa ordem, e podem criar ou modificar documentos quando necessário.

Instruções gerais:

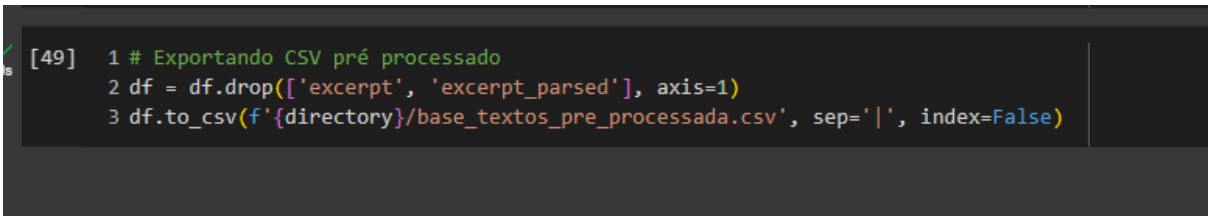
Configure o caminho da pasta correspondente ao seu drive. Todos os colabs fazem essa conexão, e isso permite o compartilhamento de dados entre eles.



```
[ ]  1 drive_path = '/content/drive'
     2 directory = f'{drive_path}/MyDrive/Dasafio 01 - Classificação'
     3 csv_file = 'base_textos_stack.csv'
```

Preprocessamento.ipynb

Será gerado um arquivo *base_textos_pre_processada.csv,* com o resultado do pré-processamento.



```
[49]  1 # Exportando CSV pré processado
      2 df = df.drop(['excerpt', 'excerpt_parsed'], axis=1)
      3 df.to_csv(f'{directory}/base_textos_pre_processada.csv', sep='|', index=False)
```

algoritmo_genetico.ipynb

Na etapa de seleção de aptidão, é importante notar que a função *get_model* pode retornar modelos diferentes.

```python
def get_model(clf_name, hyperparams=None):
  model = None
  if clf_name == 'Random Forest':
    if hyperparams != None:
      model = RandomForestClassifier(**hyperparams, random_state=random_state)
    else:
      model = RandomForestClassifier(random_state=random_state)
  elif clf_name == 'Gradient Boosting':
    if hyperparams != None:
      model = GradientBoostingClassifier(**hyperparams, random_state=random_state)
    else:
      model = GradientBoostingClassifier(random_state=random_state)
  elif clf_name == 'Decision Tree':
      if hyperparams != None:
        model = DecisionTreeClassifier(**hyperparams, random_state=random_state)
      else:
        model = DecisionTreeClassifier(random_state=random_state)
  elif clf_name == 'Hist Gradient Boosting':
      if hyperparams != None:
        model = HistGradientBoostingClassifier(**hyperparams, random_state=random_state)
      else:
        model = HistGradientBoostingClassifier(random_state=random_state)

  return model
```

Para executar o algoritmo genético, com um modelo específico, altere a variável *model_name* definida na etapa de execução do algoritmo, copiando o nome de um dos *ifs* da função get_model.

```python
# Parâmetros
population_size = 500
generations = 50
num_params = pd.DataFrame(features_train).shape[1]
mutation_rate = 0.2
inversion_rate = 0.3
parents_rate = 0.2
model_name = 'Random Forest'
random_state = 42
```

Ao final de cada execução do algoritmo genético, os resultados são adicionados ao arquivo results.csv (sendo editado, ou criado caso não exista). Assim, é armazenado o resultado da execução para cada modelo.