Irony Detection In English Tweets

André Malheiro up201706280@fe.up.pt
Diogo Gomes up201806572@fe.up.pt
Rúben Almeida up201704618@fe.up.pt
FEUP MIEIC IART Grupo 43 2020/2021

Especificação do Problema e Referências Bibliográficas

Especificação do problema:

Irony Detection in English Tweets está dividido em duas tarefas. Ambas relacionadas com a capacidade de detetar ironia em tweets em língua inglesa.
Este problema foi inicialmente apresentado numa competição de Machine Learning, a SemEval 2018.
Na primeira tarefa, ***task1,*** **é pretendido detetar se um tweet é ou não irónico.** Na segunda,***task2,*** **o objetivo é alongado à deteção do tipo de ironia.**
Este problema insere-se na categoria de problemas de aprendizagem supervisionada. Para além desta classificação, trata-se de um clássico problema de NLP. Neste tipo de problema a dificuldade reside em dois elementos:

- Filtrar o dataset de treino de tokens inúteis. Pré Processamento.
- Encontrar e escolher os parâmetros corretos entre os diferentes algoritmos de ML.

Referências Bibliográficas:

Apesar da temática da deteção de ironia ser muito específica, ela insere-se num tipo de problemas para o qual existe imensa literatura associada. A análise de sentimentos. São inúmeros os recursos sobre a temática, incluindo de colegas de anos mais avançados.

Os materiais confiáveis e utilizados por nós que merecem realce são os livros:

- Igual, Dra. Laura- Introduction DataScience Python Book
- Bird, Steven - Natural Language Processing with Python

O Dataset

O dataset usado no projeto é o original providenciado pela organização da SemEval aos participantes. A organização disponibiliza publicamente o dataset para fins educativos. No repositório com os dados a própria **fez uma divisão clara entre o train e o test dataset.** Como a competição já está finalizada temos acesso ao set de treino com os dados devidamente etiquetados.

Esta divisão em dados de treino e teste é crítica para deduzir os valores de performance e para avaliar efetivamente a qualidade do modelo.
Por essa razão, decidimos na ***task 1* usar a divisão tal como ela está.** Na *task 2* fizemos uso das **técnicas de model split e cross validation** fornecidas pela biblioteca scikit learn.

Após analisar os resultados concluímos que os dados etiquetados como test pela organização têm pouco valor e perturbam as métricas finais. Isto pode indiciar que a organização decidiu "dificultar a vida" aos participantes. Usar só os dados de treino com split e técnicas de cross validation produziu melhores métricas

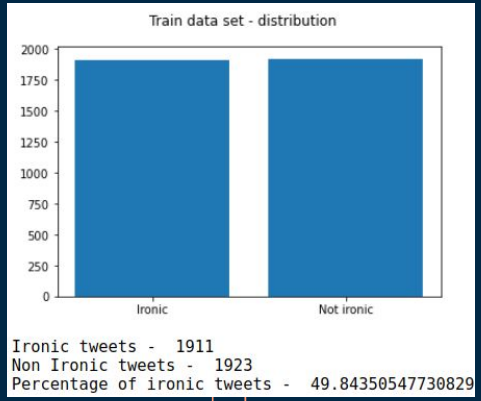


Fig.1: Distribuição dos tweets de train por labels.

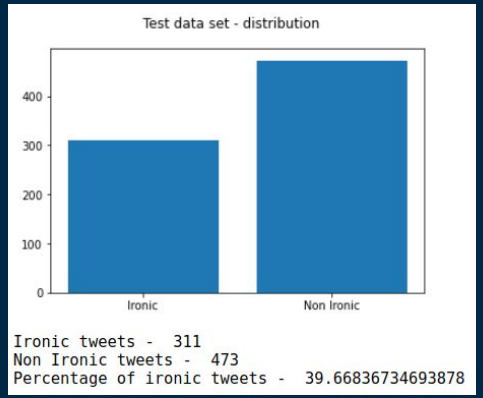


Fig.2: Distribuição dos tweets de test por labels.

Bibliotecas e Algoritmos Utilizados

Fazendo uso da biblioteca scikit learn podemos executar uma gama muito variada e completa de algoritmos de ML. Para além das 3 bibliotecas requeridas no trabalho, introduzimos um total de seis algoritmos.

Os seis algoritmos:

Naïve Bayes **Decision Trees** **Neural Networks** **KNN** **SVM** **Random Forest**

No caso de algoritmos como as decision trees que permitem usar a técnica de Grid Search, ela foi explorada, de forma a encontrar os melhores parâmetros para executar o modelo.

A linguagem de programação utilizado será Python3 e é usado o ambiente Jupyter Notebook usando como suporte o anaconda para gestão de dependências de Python.

Para além de utilizar o Scikit learn, iremos utilizar outras bibliotecas como:

- Pandas: Para carregar e manipular os dados de uma forma estruturada
- Numpy: Requisitada pelo Pandas para operações sobre arrays
- NLTK: Bibliotecas de NLP que introduzem os Stemmers e os Lemmatizers
- **contractions:** Biblioteca de Python que expandem os vocábulos ingleses que recebem apóstrofe para serem encurtados
- Seaborn e Matplotlib: Para poder fazer gráficos da matriz de confusão de forma user-friendly
- **Emoji:** Biblioteca externa que permite desmembrar emojis em palavras, tornando-os úteis para análise lexical

Pré processamento

De forma a ter sido possível obter os melhoramentos apresentados foi necessário aplicar uma cadeia de pré processamentos nos tweets de forma a normalizar os tweets. Para além dessas modificações foi aplicado a biblioteca contractions, **bem como Stemming, Lemmatization** e posterior vetorizando usando a medida TF-IDF, utilizando a biblioteca NLTK e Scikit Learn.

Pré processamentos introduzidos:

- Conversão para lowercase.
- Remoção das mentions de twitter. Palavra que começa por "@".
- Remoção do "#" nas Hashtags.
- Remoção de algarismos.
- Remoção de links HTTP.
- Desmembramento dos emojis em keywords. E normalização posterior destas keywords.

Vetorização e Tamanho da Gramática : O Dilema

TF-IDF ou Word counter?

Scikit learn a biblioteca usada para criar a bag of words introduz 2 formas de o fazer, usando uma simples contagem de palavras ou a medida TF-IDF. A análise revelou-se inconclusiva, não existiu diferença significativa. TF-IDF, em tese melhor é mais vocacionada para documentos extensos, no nosso projeto abordamos tweets, que no máximo só poderão conter 180 caracteres. Usamos TF-IDF.

Tamanho do vocabulário?

Um dos dilemas que o grupo enfrentou foi a estabilização da performance dos modelos mediante o tamanho do vocabulário usado. Em tese era ideal que pudéssemos utilizar o máximo de vocábulos possível pois isso iria enriquecer o poder da nossa linguagem.

Vocabulários muito extensos, generalizam em demasia o modelo, dispersando-o das *keywords*, mais relevantes. Um vocabulário parco, é incapaz de detetar traços textuais fundamentais. **Para a primeira task** estabelecemos o tamanho ótima a orbitar **em torno de 800 palavras.** Na segunda task, uma análise do mesmo gênero determinou que o valor de vocabulário produzia melhores resultados com **7500 vocábulos**.

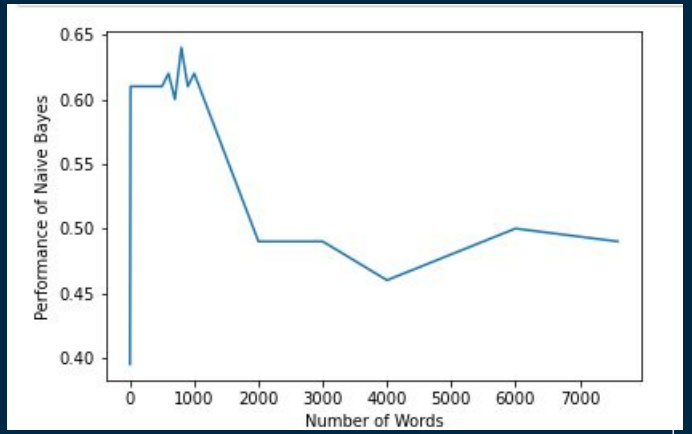


Fig.3 : Naive Bayes performance mediante número de vocábulos

Task 1: Ironic vs. non-ironic I/II

Na *task 1*, o objetivo é definir modelos capazes de detectar se um tweet é ou não irônico. Esta classificação é uma simples classificação binária.

Tal como referido fizemos uso da divisão do dataset providenciada pela organização. Esta decisão não se revelou muito acertada.

Comparação da curva ROC dos Diferentes Modelos:

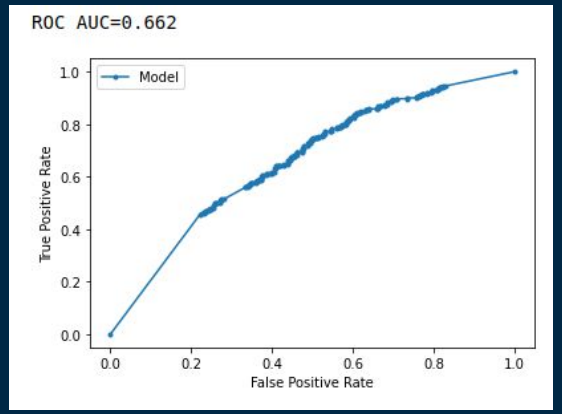


Fig.4 : Curva ROC para Naive Bayes

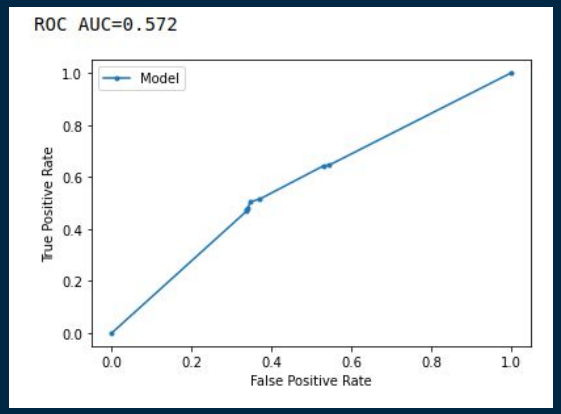


Fig.5 : Curva ROC para Decision Tree

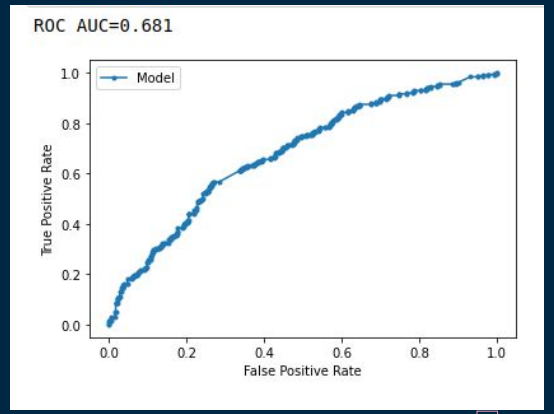


Fig.6 : Curva ROC para Neural Network

Task 1: Ironic vs. non-ironic II/II

Comparação da curva ROC dos Diferentes Modelos:

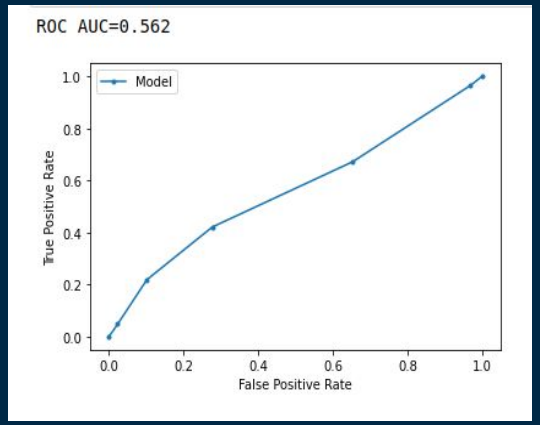


Fig.7 : Curva ROC para KNN

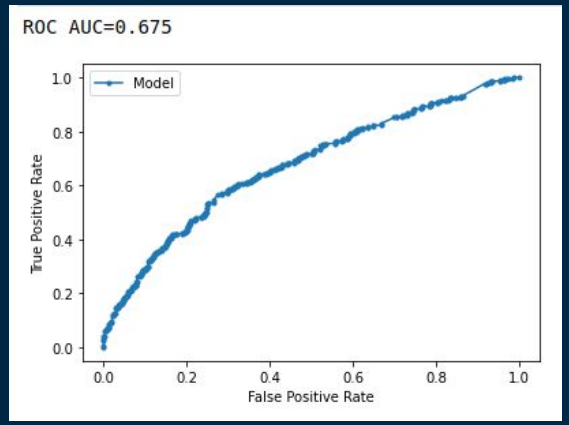


Fig.8 : Curva ROC para SVM

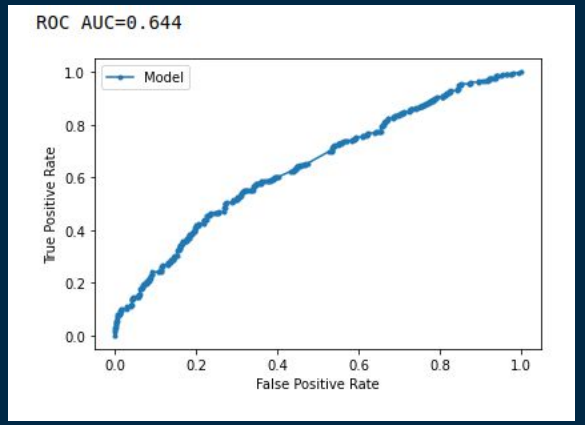


Fig.9 : Curva ROC para Random Forest

Análise dos resultados da Task 1:

- A divisão proposta pela organização não promove um bom resultado dos modelos
- Os modelos que produziram melhores resultados foram, como esperado, os SVM e as Neural Networks
- O pré processamento dos dados permitiu ganhos razoáveis de 15%
- **De forma a melhorar os resultados, mais e melhores dados são necessários.**

Task 2 – Which Type of Irony? I/II

Na *task 2*, o objetivo é encontrar modelos capazes de identificar se um tweet em inglês é irónico ou não, e também, que tipo de ironia ele compreende. Desta forma, *task 2*, é um problema de classificação do tipo **multiclass.**

Nesta task, o dataset usado foi somente o dataset de train, usando técnicas de split e de cross validation incluídas no scikit learn para evitar overfitting. As métricas de desempenho foram bastante melhores.

Aplicaram-se as mesmas estratégias de pré-processamento usadas na *task 1*.

Task 2, dado o seu dataset estar altamente desbalanceado, obrigou a aplicar técnicas de over sampling, esta estratégia sem qualquer *tunning* adicional permitiu ganhos de 20% nos principais parâmetros avaliados.

Análise dos resultados da Task 2:

- A técnica de Oversampling permitiu logo à priori ganhos de 20%. Foi uma boa escolha.
- Tal como esperado, usando o model split e descartando o dataset de test, os modelos produziram melhores métricas, no entanto isto é normal, pois passamos a estar permeável a questões de overfitting com maior facilidade.
- **A Neural Network produziu um resultado sólido e muito equilibrado**

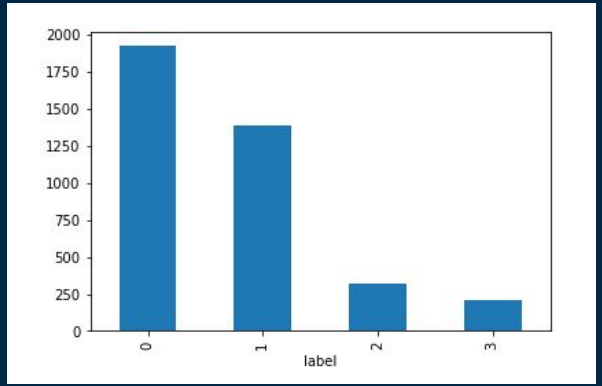


Fig.10 : Dataset desbalanceado da Task2

Task 2 – Which Type of Irony? II/II

Comparação da matriz de confusão da neural network:

As neural networks na task 2 produziram um resultado acima dos esperado:

- **Accuracy 0.891148**
- **Precision 0.888926**
- **F1-Value 0.889498**

Estes valores mais inflacionados face à task 1 demonstram claramente que o uso da divisão dos dados fornecida pela organização não é uma boa escolha.

Para além disso, de compreender ainda que é natural este resultado, visto que passamos a estar permeáveis a overfitting, no entanto, é um excelente resultado de teste.

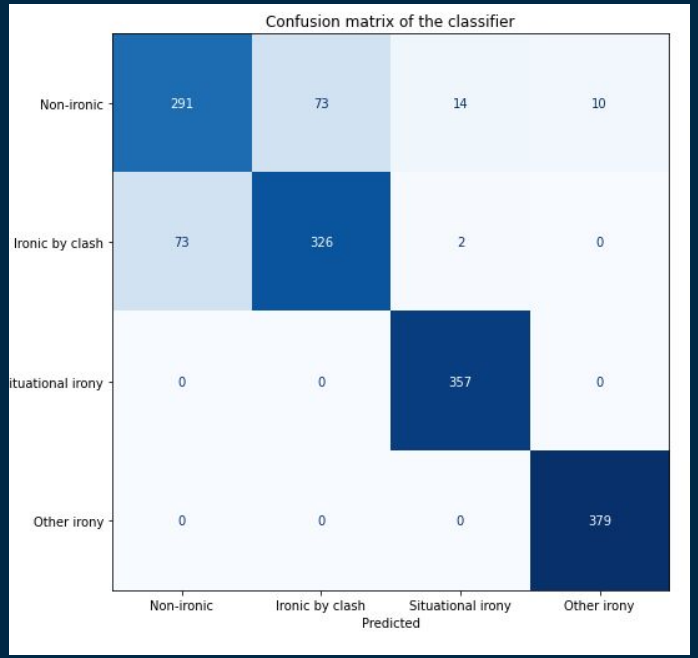


Fig.11 : Matriz de confusão para Neural Network

Conclusão

Neste projeto apesar dos conhecimentos limitados de uma área tão complexa como NLP, conseguimos explorar vários modelos, enfrentamos as dificuldades típicas deste tipo de problemas e conseguimos alcançar resultados razoáveis.

Na *task2*, termos seguido uma estratégia de oversampling guiou-nos a bons resultados.

O nosso pré processamento revelou-se útil, tendo originado **ganhos de 15-20%** de performance dos modelos.

Tal como esperado o método por eleição para obter melhores resultados são as neural networks e SVM. Naives Bayes também apresenta bons resultados, dado que é o mais rápido dos modelos a executar.

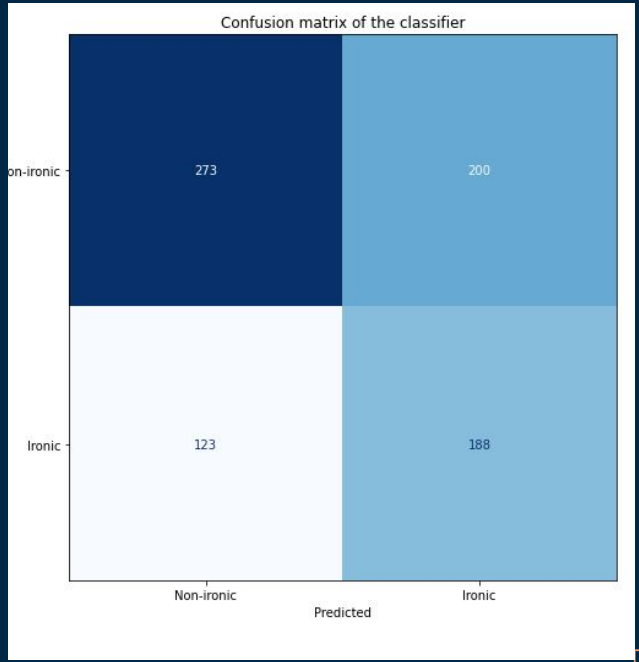
Para trabalho futuro é relevante garantir a qualidade dos dados à priori através de análises mais profundas. Para além disso, de forma a obter resultados excelentes ficou claro que 4000 mil tweets não são suficientes.

```
Enter Tweet: AI is so powerful
Non Ironic Tweet
```

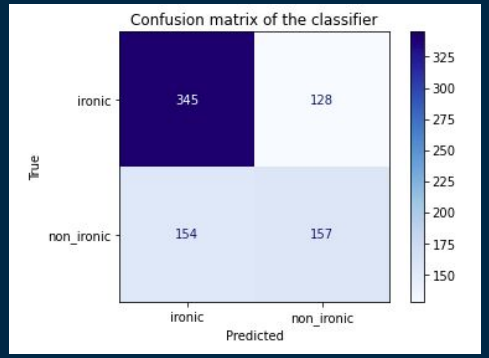
Anexo 1: Aplicação de oversampling na Task 1:

Visto que a Task 2 produziu tão bons resultados, decidimos tentar a nossa sorte e aplicar oversampling à abordagem seguida na task 1. Não teve relevância, como piorou os resultados, pois aumentamos o overfitting aos dados de treino.

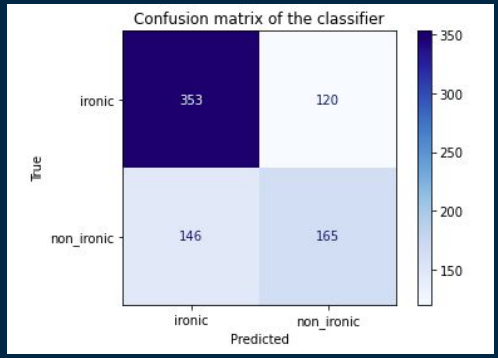
- **Accuracy 0.588010**
- **Precision 0.608130**
- **F1-Value 0.592449**



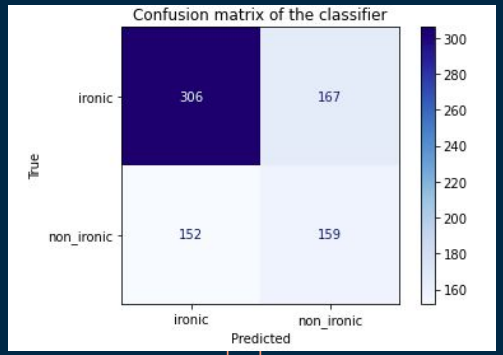
Anexo 2: Matrizes de Confusão Task 1 I/II



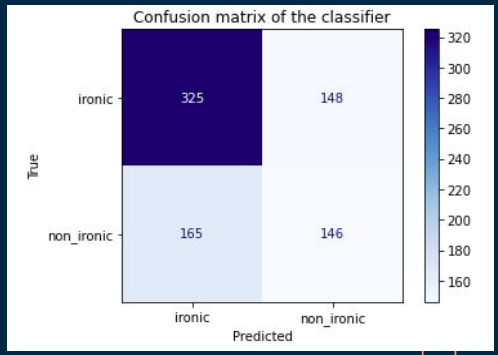
Matriz de confusão para Naive Bayes (task 1)



Matriz de confusão para Neural Network (task 1)

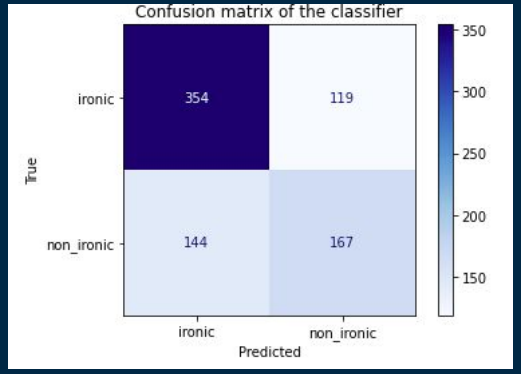


Matriz de confusão para Decision tree (task 1)

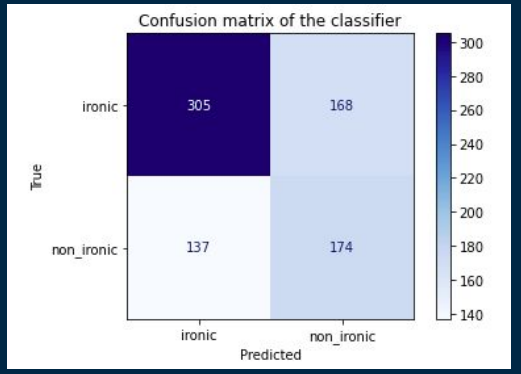


Matriz de confusão para K-Nearest Neighbors (task 1)

Anexo 2: Matrizes de Confusão Task 1 II/II

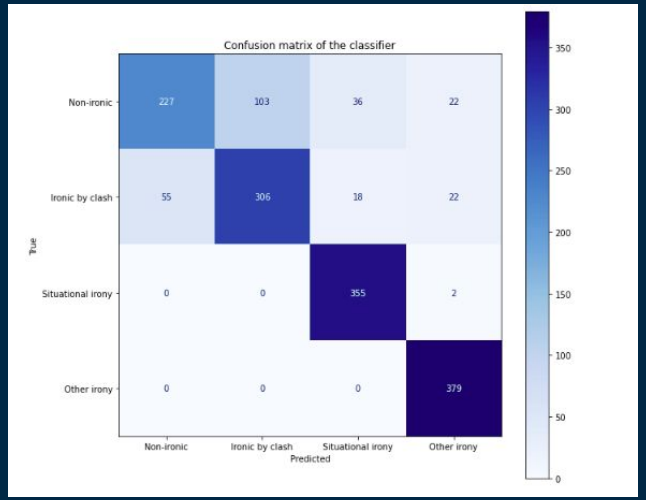


Matriz de confusão para SVM (task 1)

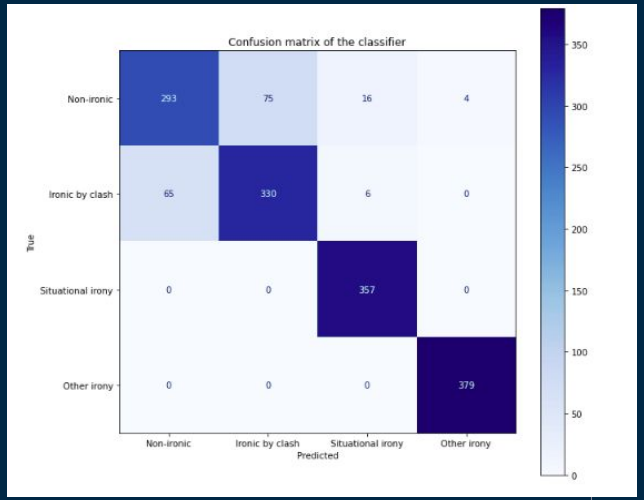


Matriz de confusão para Random Forest (task 1)

Anexo 2: Matrizes de Confusão Task 2 I/II

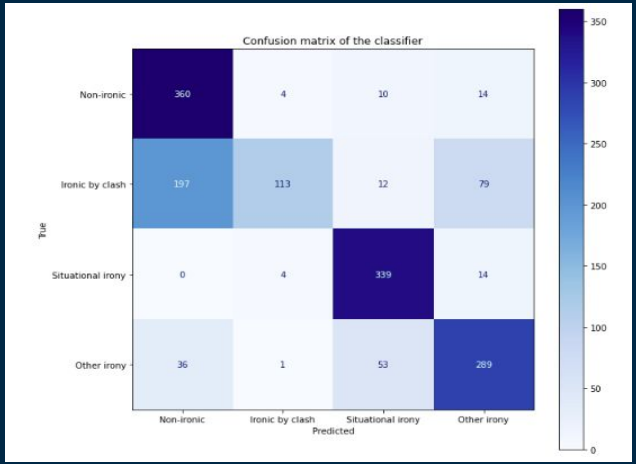


Matriz de confusão para Naive Bayes (task 2)



Matriz de confusão para Neural Network (task 2)

Anexo 2: Matrizes de Confusão Task 2 II/II



Matriz de confusão para Decision Tree (task 2)