Université Mohammed V de Rabat
École Nationale Supérieure d'Informatique et d'Analyse des Systèmes

Report :
Branch : Artificial Intelligence

Propagation of false information in a social network

Achieved by :
ESSAHEL Rim

Supervised by :
Mr. LAZAAR Mohamed

Academic year : 2021-2022

Acknowledgement

This project has been for me a chance to open up to a new deep field, and thanks to it I have been enriched by information and I have realized the importance of the analysis of social networks in real world applications. Although at the beginning it seemed difficult to understand and I met the problem of the lack of documentation, but this has not prevented me from gaining a lot of information that will be very useful to me in my course and for this I would like to thank you Professor Mr. LAZAAR Mohamed for this project .

Résumé

Ce projet effectué dans le cadre du projet du module analyse des réseaux sociaux, ce projet intitulé : "Propagation des fausses informations dans un réseau social" vise à analyser la propagation des fausses informations et aussi prédire le nombre des retweets d'un tweet qui contient une fausse information.

Afin d'aboutir à cet objectif, on a commencer par chercher les données, les nettoyer, jusqu'à le déploiement du modèle.

Abstract

This project carried out within the context of the project of the module analysis of social networks, this project entitled : "Propagation of false information in a social network" aims to analyze the propagation of false information and also predict the number of retweets of a tweet that contains false information.

In order to achieve this objective, this work started by searching the data, cleaning them, until the deployment of the model.

Table des matières

Acknowledgement **1**

Résumé **2**

Abstract **3**

LIST OF FIGURES **5**

Introduction **6**

1 General approach of the project **7**

Introduction 7

1.1 project planning 7

1.2 problematic and approach 8

Conclusion 8

2 Theoretical part **9**

2.1 Descision Trees 9

2.2 Random Forest 11

3 Realization **13**

3.1 Tools 13

3.2 implementation 14

3.2.1 Reading Data : 14

3.2.2 Analysis : 15

3.2.3 Machine learning model : 16

3.3 Results and Evaluation 17

3.4 deployment : 18

Conclusion and Discussion **20**

References **21**

Table des figures

1.1 Gantt chart 7

2.1 Decision trees 10
2.2 Random Forest 11

3.1 Python 13
3.2 Jupyter notebook 13
3.3 Networkx 14
3.4 Kedro 14
3.5 Reading data 15
3.6 Analysis 15
3.7 Creating our Data 16
3.8 Removing outliers 16
3.9 Machine learning Model 17
3.10 Table : T/F P/N 17
3.11 Evaluation metrics 17
3.12 Model Evaluation 18
3.13 Deployment 18
3.14 Results 1 19
3.15 Results 2 19

Introduction

The term "fake news" has always been widely used by the news media, in social media. With so much free information online, it's easy to get distracted by all the fake material polluting the web. Authoritative resources that were once considered official have been invaded by online platforms intentionally created to share and disseminate false and low-quality information.

This report presents in a simple and detailed way all the work done throughout the project. At the beginning I will present the general approach of the project, then we will approach the theory of the used machine learning model, in the following chapter we will see the stages of the realization of the project as well as the obtained results, finally we will make a conclusion or we will present a new problematic.

Chapitre 1

General approach of the project

Introduction :

The general context of the project aims to present the problem and the approach chosen to address it. This chapter introduces the planning scheme of the project stages, then details the problematic, and finally concludes with the exposition of the project management methodology adopted.

1.1 project planning

The gantt chart shows the different steps to finalize the project according to time :

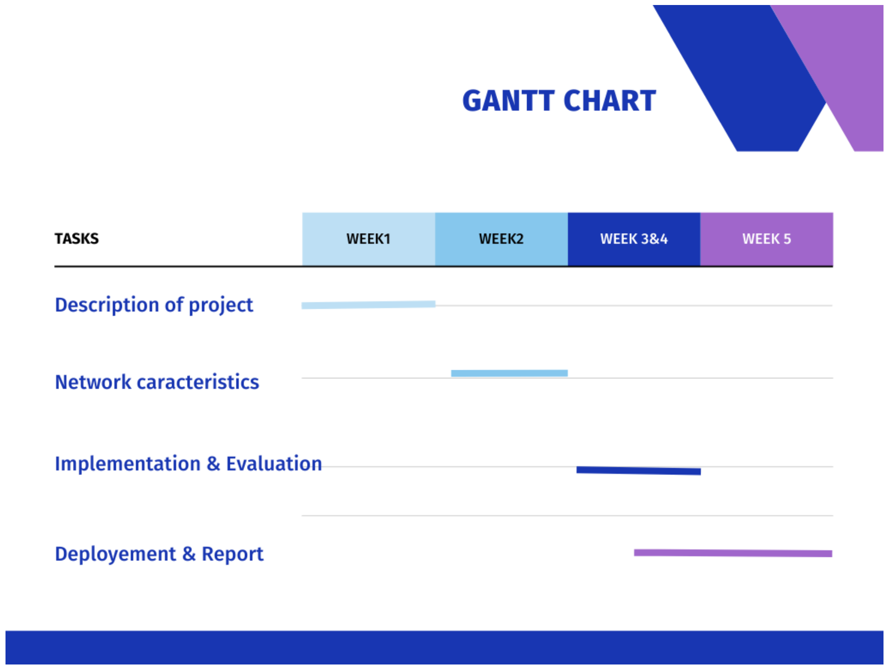


FIGURE 1.1 – Gantt chart

1.2 problematic and approach

The problem that this project deals with is the propagation of false information in the famous social network twitter. It turns out that there are people who share false information about famous people or politics, for this we must measure the impact of false information by following the retweets of people and know the characteristics that help the spread of this information for example accounts with many active subscribers, etc. ... And so we can predict the number of retweets or also the number of people who have seen this false information.

Conclusion :

The purpose of this introductory chapter is to familiarize ourselves with the global context of the project, by presenting the problems and issues raised by the situation, and by highlighting the work methodology followed and the steps taken in order to carry out a project of this scope.

Chapitre 2

Theoretical part

In this chapter we will approach the theory of the algorithm used in this project, then we will present the random forest algorithm and since it is based on decision trees we will present them both.

2.1 Descision Trees

A decision tree is a flowchart-like tree structure where an internal node represents a feature, the branch represents a decision rule, and each leaf node represents the outcome. The highest node in a decision tree is called the root node. It learns to partition based on the value of the attribute. It partitions the tree recursively, which is called recursive partitioning.

The basic idea of decision tree algorithm is as follows :

- Select the best attribute using attribute selection measures (ASM) to divide the records.
- Make this attribute a decision node and divide the data set into smaller subsets.
- Start building the tree by repeating this process recursively for each child until one of the conditions matches :
  1. All tuples belong to the same attribute value.
  2. There are no more attributes left.
  3. There are no more instances.

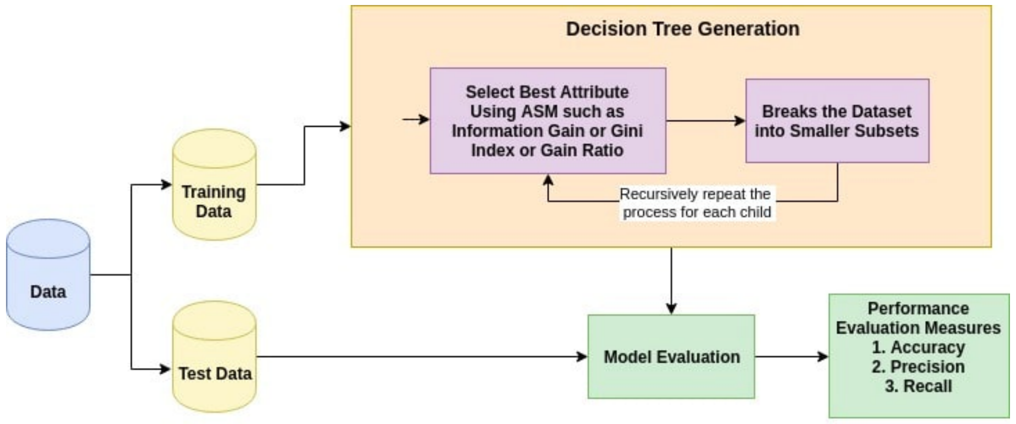

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures 1. Accuracy 2. Precision 3. Recall

FIGURE 2.1 – Decision trees

Attribute Selection Measures :

The attribute selection measure is a heuristic for selecting the split criterion that partitions the data in the best possible way. It is also known as split rules because it helps us determine the breakpoints of tuples on a given node. ASM provides a rank to each feature (or attribute) by explaining the given data set. The attribute with the highest score will be selected as the separating attribute (Source). In the case of a continuously valued attribute, the separation points for the branches must also be defined. The most popular selection measures are information gain, gain ratio and Gini index.

Information Gain :

$$\text{Info(D)= - } \sum_{i=1}^{m} pi \log_2 pi$$

Gain Ratio :

$$Gain\ Ratio = \frac{Information\ Gain}{SplitInfo} = \frac{Entropy\ (before) - \sum_{j=1}^{K} Entropy(j,\ after)}{\sum_{j=1}^{K} w_j \log_2 w_j}$$

Gini index :

$$\text{Gini(D)}=1-\sum_{i=1}^{m} \text{Pi}^2$$

Where, pi is the probability that a tuple in D belongs to class Ci.

Advantages : It can easily capture Non-linear patterns. It requires fewer data preprocessing from the user, for example, there is no need to normalize columns. It can be used for feature engineering such as predicting missing values, suitable for variable selection.

Disavantages : Sensitive to noisy data. It can overfit noisy data. Decision trees are biased with imbalance dataset, so it is recommended that balance out the dataset before creating the decision tree.

2.2 Random Forest

Random forests are a supervised learning algorithm. It can be used for both classification and regression. A forest is composed of trees, and the more trees there are, the more robust a forest is. Random forests create decision trees on randomly selected data samples, get the prediction of each tree and select the best solution by voting.

Random Forest is technically is an ensemble method of decision trees generated on a randomly split dataset. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result.

It works in four steps :

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

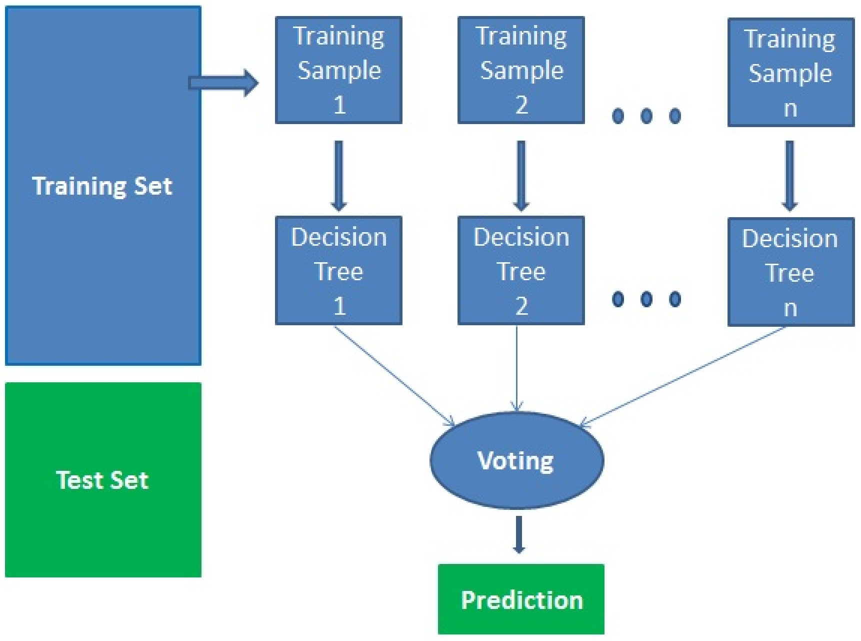


FIGURE 2.2 – Random Forest

Advantages :

— Random forests are considered a very accurate and robust method due to the number of decision trees involved in the process.

— It does not suffer from the overfitting problem. The main reason is that it takes the average of all predictions, which cancels out bias.

— The algorithm can be used in both classification and regression problems.
— Random forests can also handle missing values.

Disadvantages :

— Random forests are slow to generate predictions because they have multiple decision trees. Each time it makes a prediction, all the trees in the forest must make a prediction for the same given input, and then vote on it. This whole process takes time.
— The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path through the tree.

Chapitre 3

Realization

3.1 Tools

In this project we used the python language, it is a powerful language with the libraries that it presents especially in the field of data science :

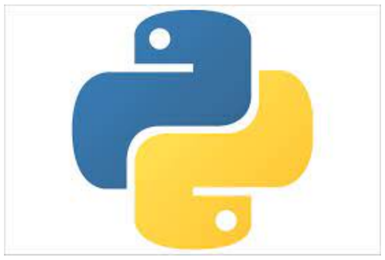


FIGURE 3.1 – Python

Jupyter Notebook : is a web-based interactive computing platform. The notebook combines live code, equations, narrative text, visualizations...

FIGURE 3.2 – Jupyter notebook

NetworkX is a Python library for the study of graphs and networks :

FIGURE 3.3 – Networkx

Kedro is an open-source Python framework for creating reproducible, maintainable and modular data science code :

FIGURE 3.4 – Kedro

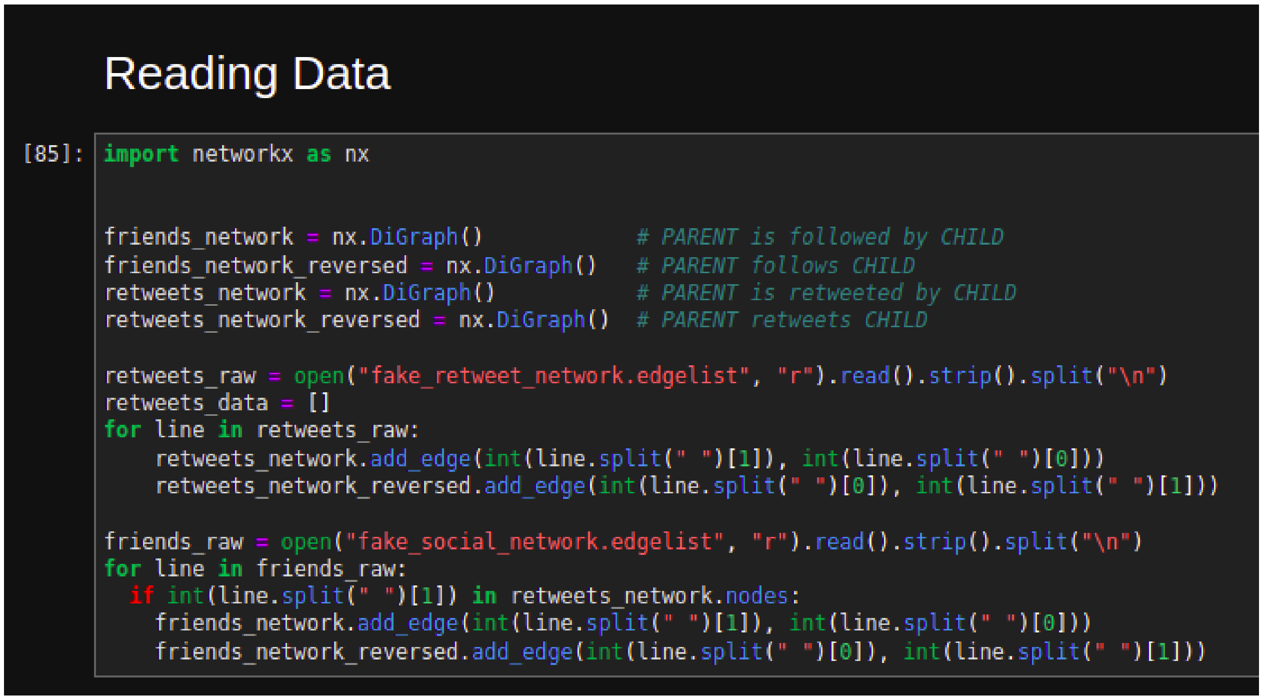
3.2 implementation

3.2.1 Reading Data :

We have two files : **fake_social_network.edgelist** and **fake_retweet_network.edgelist** , both of them contain graphs in the format **edgelist**, i.e. each line defines an arc from the first number to the second, and they are directed graphs.

-**fake_social_network.edgelist** file contains a directed graph of which follows which in twitter, with 456626 nodes, 14855842 edges.

- **fake_retweet_network.edgelist** file contains a directed graph of who retweets who, i.e. who shares false information, with 256451 nodes, 328132 edges.



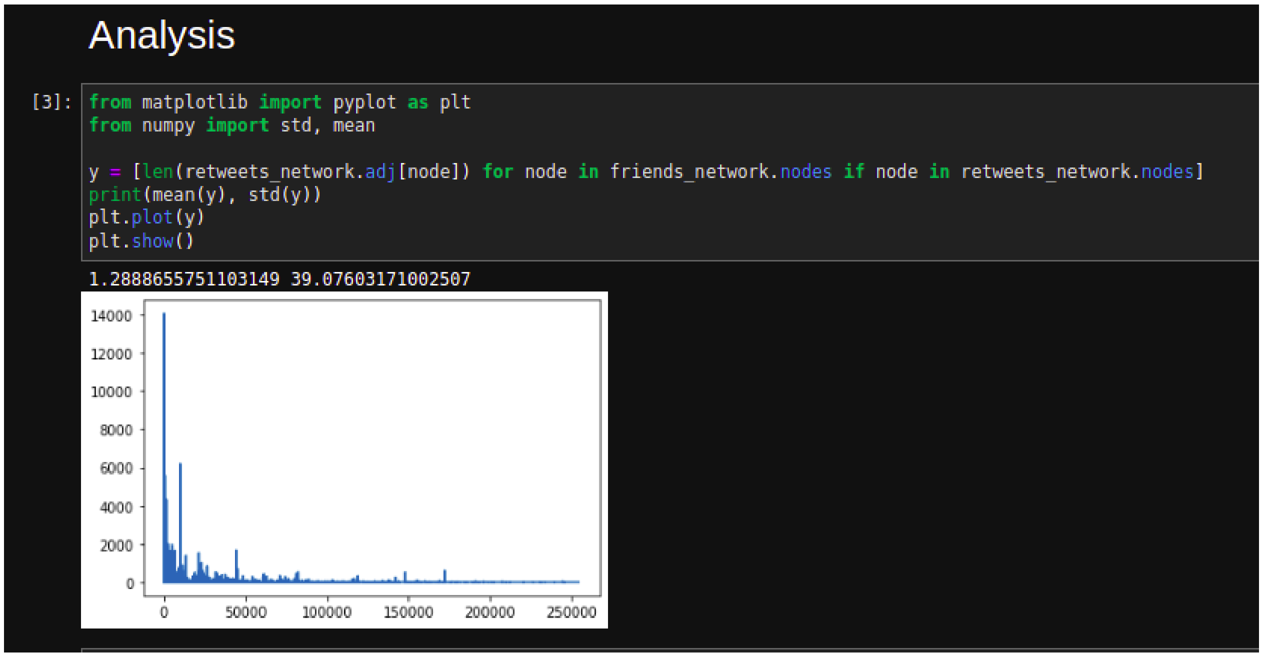
Reading Data

```
[85]: import networkx as nx


friends_network = nx.DiGraph()            # PARENT is followed by CHILD
friends_network_reversed = nx.DiGraph()   # PARENT follows CHILD
retweets_network = nx.DiGraph()           # PARENT is retweeted by CHILD
retweets_network_reversed = nx.DiGraph()  # PARENT retweets CHILD

retweets_raw = open("fake_retweet_network.edgelist", "r").read().strip().split("\n")
retweets_data = []
for line in retweets_raw:
    retweets_network.add_edge(int(line.split(" ")[1]), int(line.split(" ")[0]))
    retweets_network_reversed.add_edge(int(line.split(" ")[0]), int(line.split(" ")[1]))

friends_raw = open("fake_social_network.edgelist", "r").read().strip().split("\n")
for line in friends_raw:
  if int(line.split(" ")[1]) in retweets_network.nodes:
    friends_network.add_edge(int(line.split(" ")[1]), int(line.split(" ")[0]))
    friends_network_reversed.add_edge(int(line.split(" ")[0]), int(line.split(" ")[1]))
```

FIGURE 3.5 – Reading data

3.2.2 Analysis :

We visualized the retweets that will be our target, and we notice that the majority, either do not retweet or have a low number of retweets.



Analysis

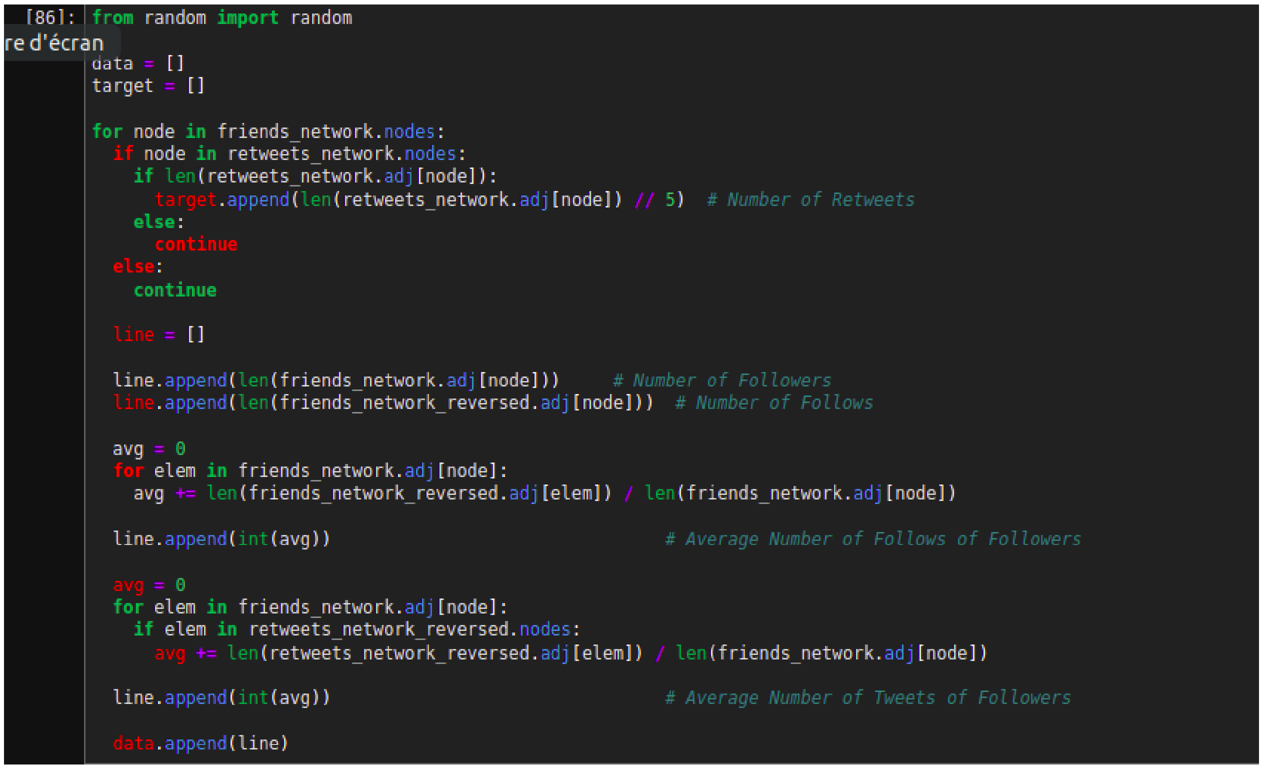
```
[3]: from matplotlib import pyplot as plt
     from numpy import std, mean

     y = [len(retweets_network.adj[node]) for node in friends_network.nodes if node in retweets_network.nodes]
     print(mean(y), std(y))
     plt.plot(y)
     plt.show()

     1.2888655751103149 39.07603171002507
```

FIGURE 3.6 – Analysis

3.2.3 Machine learning model :

To build our data on which we will apply our model, we will use the two graphs, and we will calculate the following variables :

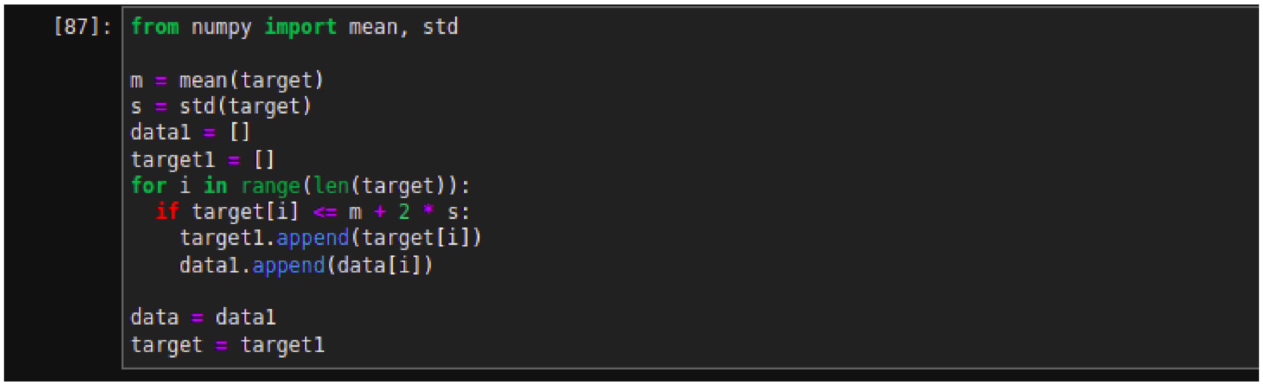
— Number of subscribers.
— Number of subscription.
— The average subscription of the subscribers of the individual. average share (retweet) of the followers.
— Target : Number of shares or retweets.



```
[86]: from random import random

      data = []
      target = []

      for node in friends_network.nodes:
        if node in retweets_network.nodes:
          if len(retweets_network.adj[node]):
            target.append(len(retweets_network.adj[node]) // 5)  # Number of Retweets
          else:
            continue
        else:
          continue

        line = []

        line.append(len(friends_network.adj[node]))      # Number of Followers
        line.append(len(friends_network_reversed.adj[node]))  # Number of Follows

        avg = 0
        for elem in friends_network.adj[node]:
          avg += len(friends_network_reversed.adj[elem]) / len(friends_network.adj[node])

        line.append(int(avg))                            # Average Number of Follows of Followers

        avg = 0
        for elem in friends_network.adj[node]:
          if elem in retweets_network_reversed.nodes:
            avg += len(retweets_network_reversed.adj[elem]) / len(friends_network.adj[node])

        line.append(int(avg))                            # Average Number of Tweets of Followers

        data.append(line)
```

FIGURE 3.7 – Creating our Data

Then we remove the outliers :



```
[87]: from numpy import mean, std

      m = mean(target)
      s = std(target)
      data1 = []
      target1 = []
      for i in range(len(target)):
        if target[i] <= m + 2 * s:
          target1.append(target[i])
          data1.append(data[i])

      data = data1
      target = target1
```

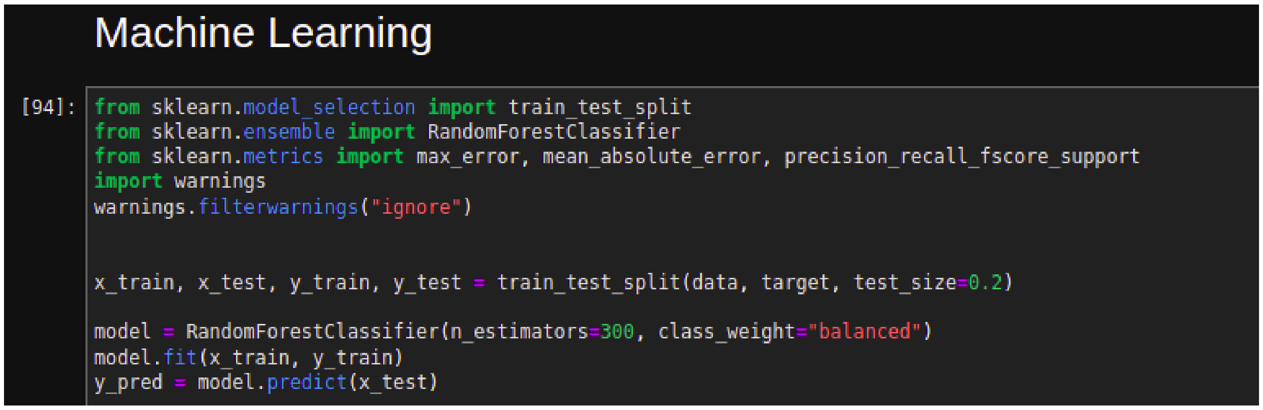
FIGURE 3.8 – Removing outliers

The model we have chosen is Random forest with the following hyperparameters :

— Number of estimators (i.e. trees of this forest) = 300.

— class_weight="balanced".

we divide our data randomly into test and training with the percentages 80 % 20 %. And we'll predict how many shares an article will get.

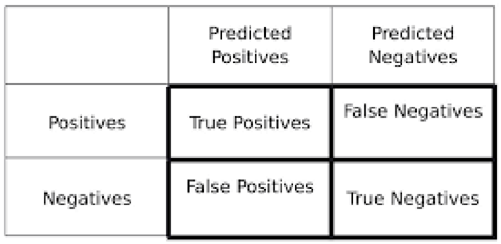


Machine Learning

```
[94]: from sklearn.model_selection import train_test_split
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.metrics import max_error, mean_absolute_error, precision_recall_fscore_support
      import warnings
      warnings.filterwarnings("ignore")


      x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.2)

      model = RandomForestClassifier(n_estimators=300, class_weight="balanced")
      model.fit(x_train, y_train)
      y_pred = model.predict(x_test)
```

FIGURE 3.9 – Machine learning Model

3.3 Results and Evaluation



	Predicted Positives	Predicted Negatives
Positives	True Positives	False Negatives
Negatives	False Positives	True Negatives

FIGURE 3.10 – Table : T/F P/N

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}}$$

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}$$

$$\text{F}_1 \text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

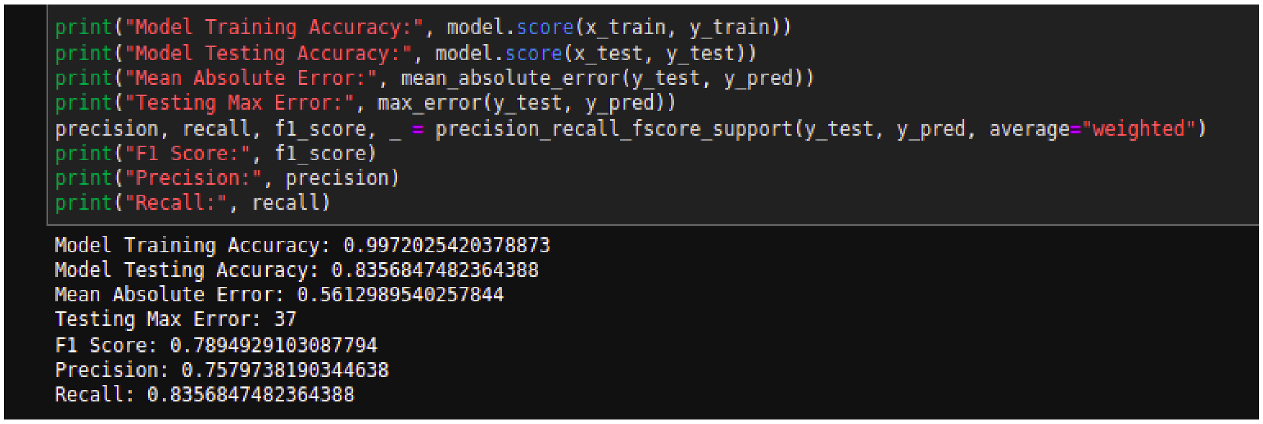
FIGURE 3.11 – Evaluation metrics

- The accuracy allows to know the proportion of good predictions compared to all predictions. The operation is simply : Number of good predictions / Total number of predictions

- The precision corresponds to the number of documents correctly attributed to class i compared to the total number of documents predicted as belonging to class i (total predicted positive).

- The recall is the number of documents correctly assigned to class i compared to the total number of documents belonging to class i (total true positive).

- The F1-Score subtly combines accuracy and recall.It is more interesting than accuracy because the number of true negatives (tn) is not taken into account. And in situations of imbalanced class as it is the case with our dataset.



```
print("Model Training Accuracy:", model.score(x_train, y_train))
print("Model Testing Accuracy:", model.score(x_test, y_test))
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Testing Max Error:", max_error(y_test, y_pred))
precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y_pred, average="weighted")
print("F1 Score:", f1_score)
print("Precision:", precision)
print("Recall:", recall)
```

```
Model Training Accuracy: 0.9972025420378873
Model Testing Accuracy: 0.8356847482364388
Mean Absolute Error: 0.5612989540257844
Testing Max Error: 37
F1 Score: 0.7894929103087794
Precision: 0.7579738190344638
Recall: 0.8356847482364388
```

FIGURE 3.12 – Model Evaluation

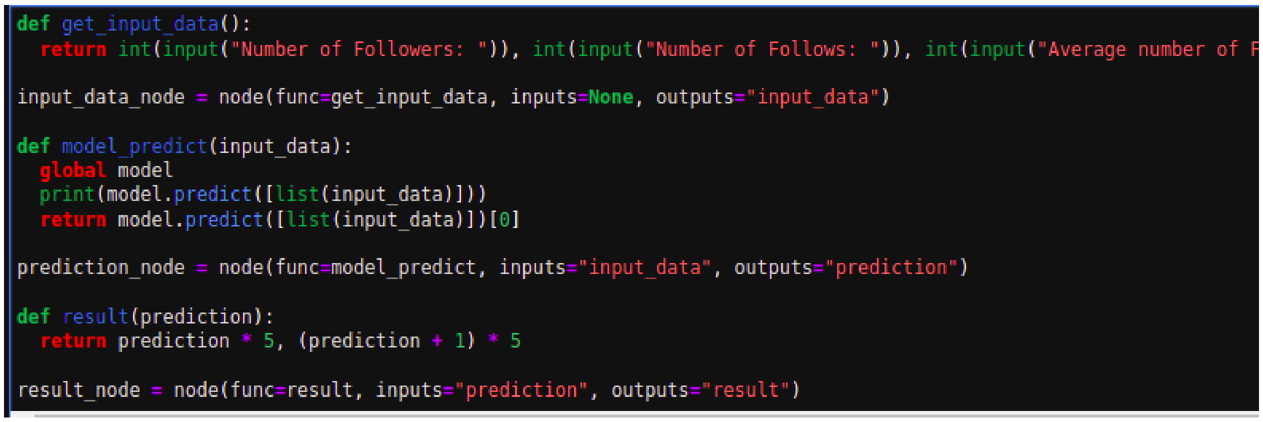
So as we can see here the model has a pretty good capacity to predict, as **accuracy training** = 0.83, as well as the **f1 score** which combines between the **Precision** and the **Recall** is pretty high, which shows that our model works well.

3.4 deployment :

The last part of the project is the deployment, we will create a pipeline from a set of nodes, which are Python functions that perform distinct, individual tasks.

The nodes we have are :

1. **input_data_node** which corresponds to the function **get_input_data** which will take the numbers we enter and return them.
2. **prediction_node** which corresponds to the function **model_predict** which will use our model to predict the output from the given inputs.
3. **result_node** which corresponds to the function **result** which will return the result.

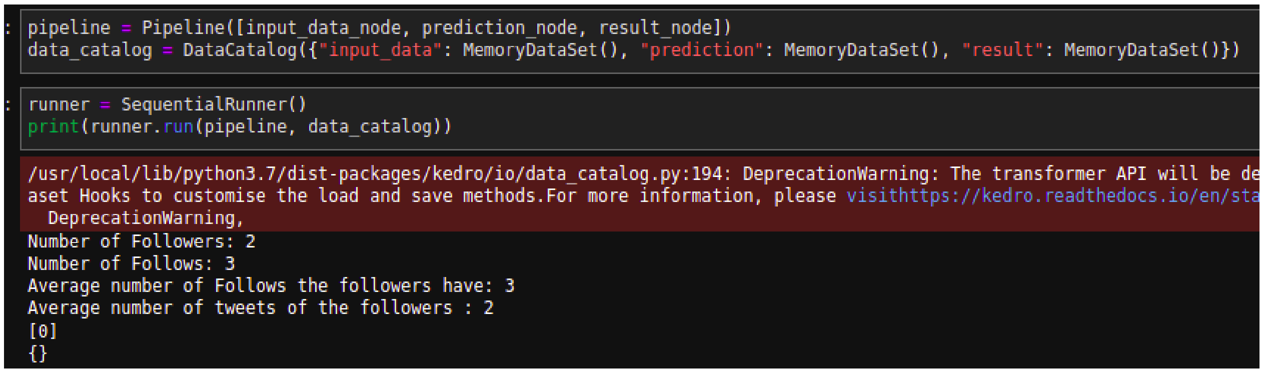


```
def get_input_data():
  return int(input("Number of Followers: ")), int(input("Number of Follows: ")), int(input("Average number of F

input_data_node = node(func=get_input_data, inputs=None, outputs="input_data")

def model_predict(input_data):
  global model
  print(model.predict([list(input_data)]))
  return model.predict([list(input_data)])[0]

prediction_node = node(func=model_predict, inputs="input_data", outputs="prediction")

def result(prediction):
  return prediction * 5, (prediction + 1) * 5

result_node = node(func=result, inputs="prediction", outputs="result")
```

FIGURE 3.13 – Deployment

Now we will build our pipeline by specifying the inputs and outputs for each of its nodes. And finally we will execute Kedro, choosing to execute the nodes sequentially, i.e. one node will execute after the other.

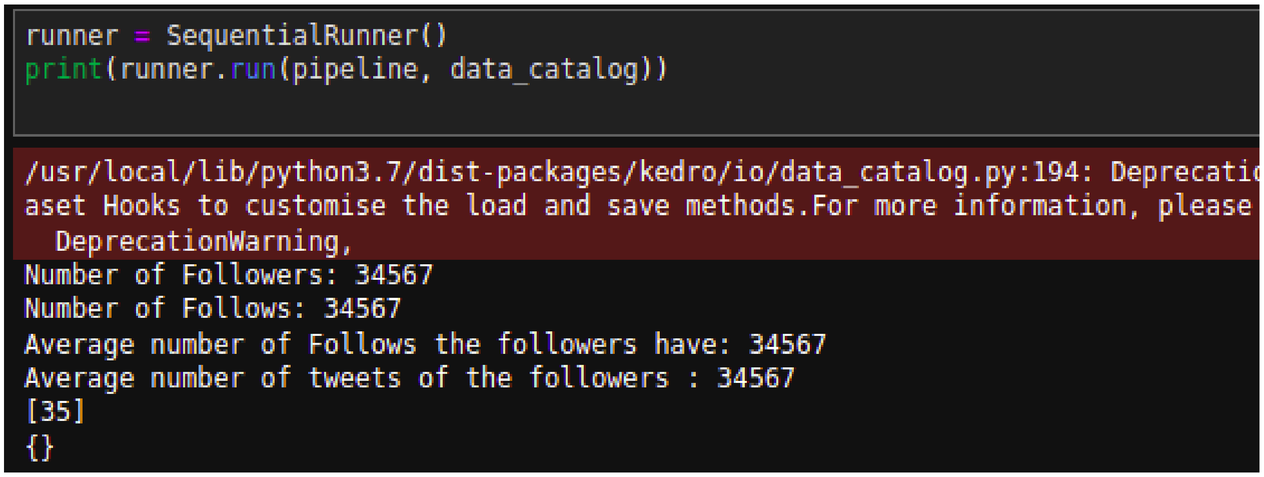
Now we will use our model and predict how many retweets of a false information we will have with a weak account, we have 0 retweet as expected :



```
: pipeline = Pipeline([input_data_node, prediction_node, result_node])
  data_catalog = DataCatalog({"input_data": MemoryDataSet(), "prediction": MemoryDataSet(), "result": MemoryDataSet()})

: runner = SequentialRunner()
  print(runner.run(pipeline, data_catalog))

  /usr/local/lib/python3.7/dist-packages/kedro/io/data_catalog.py:194: DeprecationWarning: The transformer API will be de
  aset Hooks to customise the load and save methods.For more information, please visithttps://kedro.readthedocs.io/en/sta
    DeprecationWarning,
  Number of Followers: 2
  Number of Follows: 3
  Average number of Follows the followers have: 3
  Average number of tweets of the followers : 2
  [0]
  {}
```

FIGURE 3.14 – Results 1

Now we will try with an very active account, we obtain 35 retweets :



```
runner = SequentialRunner()
print(runner.run(pipeline, data_catalog))

/usr/local/lib/python3.7/dist-packages/kedro/io/data_catalog.py:194: Deprecati
aset Hooks to customise the load and save methods.For more information, please
  DeprecationWarning,
Number of Followers: 34567
Number of Follows: 34567
Average number of Follows the followers have: 34567
Average number of tweets of the followers : 34567
[35]
{}
```

FIGURE 3.15 – Results 2

Conclusion and Discussion

To conclude, in this project, and in order to carry it out, the solution that we have proposed lies in the planning. We started by studying the subject, determining the problem and specifying the objectives, and finally we resorted to the realization. I feel that this project has been beneficial to me, it has been a real opportunity to discover new tools and approaches. Now how can we use artificial intelligence to limit the spread of false information and warn users of this information ?

References

https://medium.com/quantumblack/getting-started-with-kedro-67edcc316f6a

https://www.datacamp.com/community/tutorials/decision-tree-classification-python

https://www.datacamp.com/community/tutorials/random-forests-classifier-python

https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-per