
# Fake Covid News Detection

CS 415: Machine Learning

Zaiyr Sharsheev

BISHKEK

2021

# Table of Contents

**Acknowledgement** 2

**Introduction** 3

**Explanations** 4

**References** 9

# ACKNOWLEDGEMENT

To begin with, I would like to say thanks to our lecturer **Mr. Selcuk Cankurt** for the knowledge that was gained during this year in the study of **Machine Learning**. And also thanks to my classmates who helped to consolidate the knowledge gained and to even improve it somewhere.

# INTRODUCTION

This project is based on developing an EPL (English Premier League) Football Matches Predictions using **Python Programming Language** and **Jupyter Notebook** (an open-source web application that allows to create and share documents that contain live code, equations, visualizations and narrative text). For that I used **Pandas** (an open source Python package that is most widely used for data science/data analysis and machine learning tasks), **Scikit-learn** (simple and efficient tools for predictive data analysis). You will see a more detailed explanation of each of them below.



# EXPLANATIONS

In this report I have given explanations of what is the purpose of this project and how this Fake Covid News Detection works. I have explained here step by step so that it will surely help to become more understandable with it. Below are my explanations:

## *What is the purpose?*

The goal of this project is to find out if a specific news about coronavirus is true or false. After the outbreak of the pandemic, some media provide false news about the coronavirus for the ratings. My mission is to help people filter news and keep abreast of only true current news.

## *Execution Procedure*

**Step №1**

```
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import pandas as pd
import re
import string
```

Figure 1. Import

I imported all the required libraries and modules that I described above. From sklear.metrics I imported classification_report. It's necessary for building a text report showing the main classification metrics. From sklearn.model_selection I imported train_test_split. It splits arrays or matrices into random train and test subsets.

**Step №2**

```
df = pd.read_csv("Constraint_Train.csv")
```

```
df.shape
```

```
(6424, 3)
```

```
df.loc[df['label'] == 'real','label'] = 1
df.loc[df['label'] == 'fake','label'] = 0
df.isnull().sum()
```

```
id       0
tweet    0
label    0
dtype: int64
```

Figure 2. Reading from csv file

Here thanks to Pandas library I read a comma-separated values (csv) file into Dataframe objects. This file I got from https://www.kaggle.com/lunamcbride24/covid19-tweet-truth-analysis/data website. You can download the Constraint_Train.csv file there and use it. I also displayed the shape of the Dataframe objects. In the column 'label' I changed 'real' value as 1 and 'fake' value as 0, because for the training model I need integer value.

## Step №3

```
def word_filter(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub("\\W"," ",text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

```
df['tweet'] = df['tweet'].apply(word_filter)
```

Figure 3. News filtering

In the column ‘tweet’ using the word_filter method, I filtered the news by removing all unnecessary special characters, numbers, links, and extra space. This will help to better train the model.

## Step №4

```
x = df['tweet']
y = df['label']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 3)

y_train = y_train.astype('int')
y_test = y_test.astype('int')
```

Figure 4. Train test

I defined dependent and independent variables as x and y and splitted the array into a train set and a test set.

## Step №5

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

Figure 5. Results of the last 10 matches

I used TfidVectorizer for converting a collection of raw text vectors.

## Step №6

### 1. Logistic Regression

```
from sklearn.linear_model import LogisticRegression

LR_model = LogisticRegression()
LR_model.fit(xv_train, y_train)

LR_model.score(xv_test, y_test)

pred_LR_model = LR_model.predict(xv_test)

print(classification_report(y_test, pred_LR_model))
```

```
              precision    recall  f1-score   support

           0       0.93      0.89      0.91       753
           1       0.91      0.94      0.92       853

    accuracy                           0.92      1606
   macro avg       0.92      0.92      0.92      1606
weighted avg       0.92      0.92      0.92      1606
```

Figure 6. Logistic Regression model

Implementing and training the Logistic Regression model, measuring the quality of the trained model.

**Step №7**

2. Decision Tree Classification

```
from sklearn.tree import DecisionTreeClassifier

DT_model = DecisionTreeClassifier()
DT_model.fit(xv_train, y_train)

DT_model.score(xv_test, y_test)

pred_DT_model = DT_model.predict(xv_test)

print(classification_report(y_test, pred_DT_model))
```

```
              precision    recall  f1-score   support

           0       0.87      0.84      0.86       753
           1       0.86      0.89      0.88       853

    accuracy                           0.87      1606
   macro avg       0.87      0.87      0.87      1606
weighted avg       0.87      0.87      0.87      1606
```

Figure 7. Decision Tree Classification model

Implementing and training the Decision Tree Classification model, measuring the quality of the trained model.

**Step №8**

3. Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier

GBC_model = GradientBoostingClassifier(random_state=0)
GBC_model.fit(xv_train, y_train)

GBC_model.score(xv_test, y_test)

pred_GBC_model = GBC_model.predict(xv_test)

print(classification_report(y_test, pred_GBC_model))
```

```
              precision    recall  f1-score   support

           0       0.89      0.86      0.88       753
           1       0.88      0.91      0.89       853

    accuracy                           0.89      1606
   macro avg       0.89      0.88      0.88      1606
weighted avg       0.89      0.89      0.89      1606
```

Figure 8. Gradient Boosting Classifier model

Implementing and training the Gradient Boosting Classifier model, measuring the quality of the trained model.

**Step №9**

### 4. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

RFC_model = RandomForestClassifier(random_state=0)
RFC_model.fit(xv_train, y_train)

RFC_model.score(xv_test, y_test)

pred_RFC_model = RFC_model.predict(xv_test)

print(classification_report(y_test, pred_RFC_model))
```

```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91       753
           1       0.91      0.93      0.92       853

    accuracy                           0.91      1606
   macro avg       0.91      0.91      0.91      1606
weighted avg       0.91      0.91      0.91      1606
```

Figure 9. Random Forest Classifier model

Implementing and training the Random Forest Classifier model, measuring the quality of the trained model.

**Step №10**

```
def check_news(n):
    if n == 0:
        return "It's A Fake Covid News"
    elif n == 1:
        return "It's Not A Fake Covid News"

def predicting(news):
    testing_news = {"text":[news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(word_filter)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred_LR = LR_model.predict(new_xv_test)
    pred_DT = DT_model.predict(new_xv_test)
    pred_GBC = GBC_model.predict(new_xv_test)
    pred_RFC = RFC_model.predict(new_xv_test)

    return print("Logistic Regression Prediction: {} \nDecision Tree Classification Prediction: {} \nGradient Boosting Classifier Predicti
        check_news(pred_LR[0]),
        check_news(pred_DT[0]),
        check_news(pred_GBC[0]),
        check_news(pred_RFC[0]))
    )
```

Figure 10. Testing using 4 models with manual entry

Testing using 4 models and forecasting - whether the news is fake or not.

**Step №11**

```
print("News №1")
news = "New variant classed of concern and named Omicron."
predicting(news)

print("================================================")

print("News №2")
news2 = "The first volunteer to take the human trial vaccine for coronavirus in the UK has died."
predicting(news2)
```

```
News №1
Logistic Regression Prediction: It's Not A Fake Covid News
Decision Tree Classification Prediction: It's Not A Fake Covid News
Gradient Boosting Classifier Prediction: It's Not A Fake Covid News
Random Forest Classifier Prediction: It's Not A Fake Covid News
================================================
News №2
Logistic Regression Prediction: It's A Fake Covid News
Decision Tree Classification Prediction: It's A Fake Covid News
Gradient Boosting Classifier Prediction: It's A Fake Covid News
Random Forest Classifier Prediction: It's A Fake Covid News
```

Figure 11. Fake Covid News Detection

# REFERENCES


1. Covid19 Tweet Truth Analysis: https://www.kaggle.com/lunamcbride24/covid19-tweet-truth-analysis/data
2. Pandas library: https://pandas.pydata.org/
3. Scikit-learn library: https://scikit-learn.org/stable/