ELG5142: Ubiquitous Sensing / Smart Cities

Assignment 2

Group 23: Assignment 2

Overview

The main objective of this assignment is to build three machine learning models to classify tasks (fake or legitimate) then make a final decision by using two ensemble frameworks.

Methodology

We followed some defined steps to obtain the aimed results:

1. Import useful packages

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
##########################################################################################
```

2. Load and Split the data

```
df = pd.read_csv('MCSDatasetNEXTCONLab.csv')
x = df.iloc[:, 0:12]
y = df.iloc[:, 12]
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42)
x_train, x_test, y_train, y_test = x_train.reset_index(drop=True), x_test.reset_index(drop=True),y_train.reset_i
```

3. Build the models

```
# models fitting and prediction
def models(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
    x_train, x_test, y_train, y_test = x_train.reset_index(drop=True), x_test.reset_index(drop=True),y_
    model = model.fit(x_train, y_train)
    y_train_pred = model.predict(x_train)
    y_test_pred = model.predict(x_test)
    accuracy_train = accuracy_score(y_train, y_train_pred)
    accuracy_test = accuracy_score(y_test, y_test_pred)
    report_test = classification_report(y_test, y_test_pred)
    return (model, y_train_pred, y_test_pred, accuracy_train, accuracy_test, report_test)
```

4. Build the first 3 models and plot its confusion matrix

```python
#-----------------------------------------MAIN--------------------------------

RF, AB, NB = RandomForestClassifier(), AdaBoostClassifier(), GaussianNB()

#first model (Random Forest)
model_RF, y_train_pred_RF, y_test_pred_RF, accuracy_train_RF, accuracy_test_RF, report_RF = models(RF, x, y)
cf_RF = ConfusionMatrix(y_test, y_test_pred_RF)
PLOT_ConfusionMatrix(cf_RF,'RF Confusion Matrix')

#second model (Adaboost)
model_AB, y_train_pred_AB, y_test_pred_AB, accuracy_train_AB, accuracy_test_AB, report_AB = models(AB, x, y)
cf_AB = ConfusionMatrix(y_test, y_test_pred_AB)
PLOT_ConfusionMatrix(cf_AB,'AB Confusion Matrix')

#third model (Naive Bayes)
model_NB, y_train_pred_NB, y_test_pred_NB, accuracy_train_NB, accuracy_test_NB, report_NB = models(NB, x, y)
cf_NB = ConfusionMatrix(y_test, y_test_pred_NB)
PLOT_ConfusionMatrix(cf_NB,'NB Confusion Matrix')
```

5. Build the Voting model manually and plot its confusion matrix

First, we have concatenated all of the test_pred Dataframes of there models together, after that we have sum the values of the rows to be something like this.

Index	0
0	3
1	3
2	3
3	3
4	1
5	3
6	3
7	3
8	3
9	3
10	3
11	3
12	3

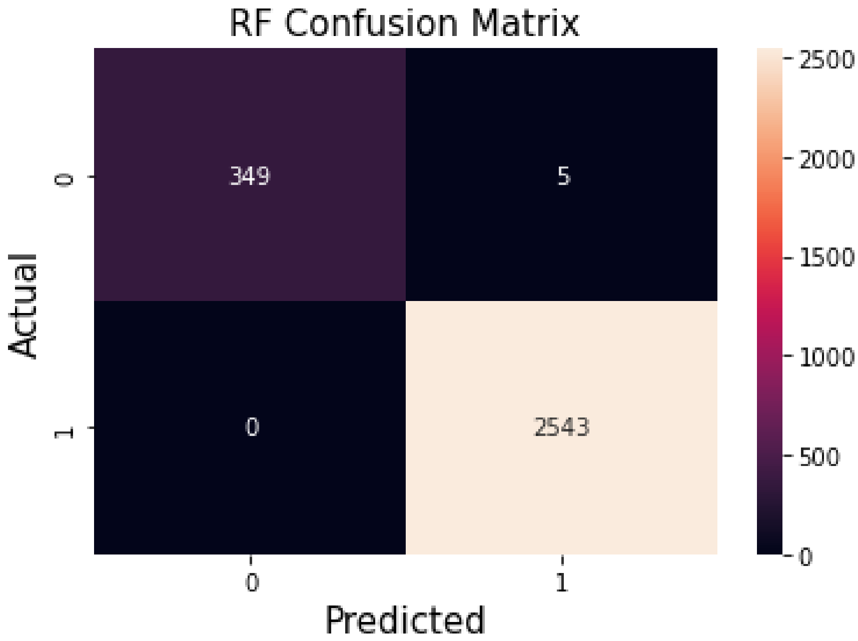
and after that we have replaced all values that equal to 1 with 0, and 3 and 2 with 1.

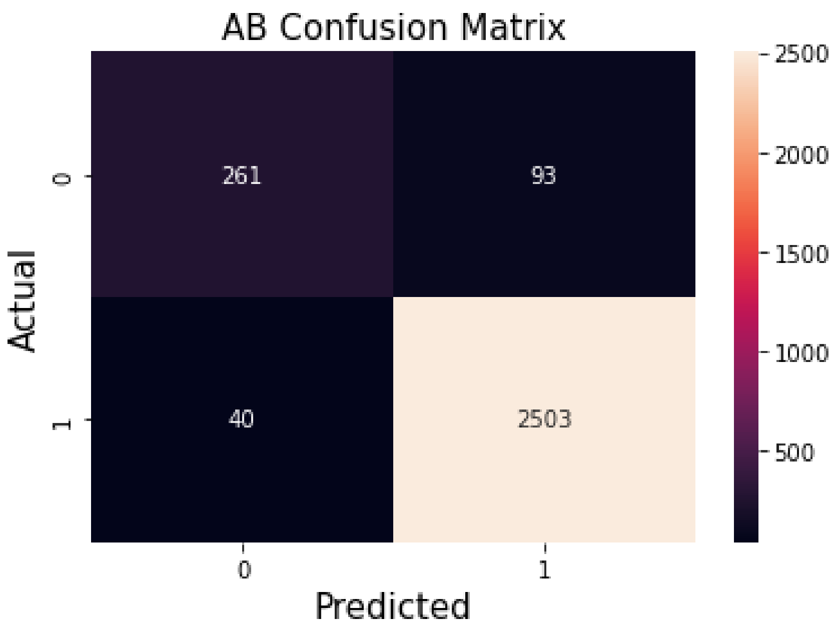
Note* 1 on the figure that is o the left mean that two models has predicted 0 on this observation and only one model predicted 1

```python
#frist ensemble framework : majority voting-based aggregator
y_pred_Voting = pd.concat([pd.DataFrame(y_test_pred_RF), pd.DataFrame(y_test_pred_AB), pd.DataFrame(y_test_pred_NB)],axis
y_pred_Voting = y_pred_Voting.sum(axis = 1)
y_pred_Voting = y_pred_Voting.replace(1, 0)
y_pred_Voting = y_pred_Voting.replace(3, 1)
y_pred_Voting = y_pred_Voting.replace(2, 1)
cf_Voting = ConfusionMatrix(y_test, y_pred_Voting)
PLOT_ConfusionMatrix(cf_Voting,'Voting Confusion Matrix')
report_Voting = classification_report(y_test, y_pred_Voting)
accuracy_Voting = accuracy_score(y_test, y_pred_Voting)
```

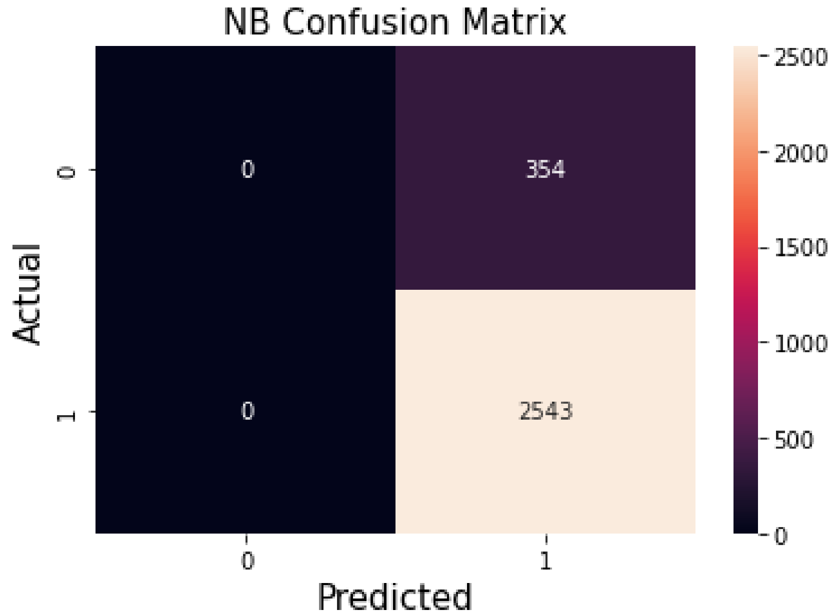
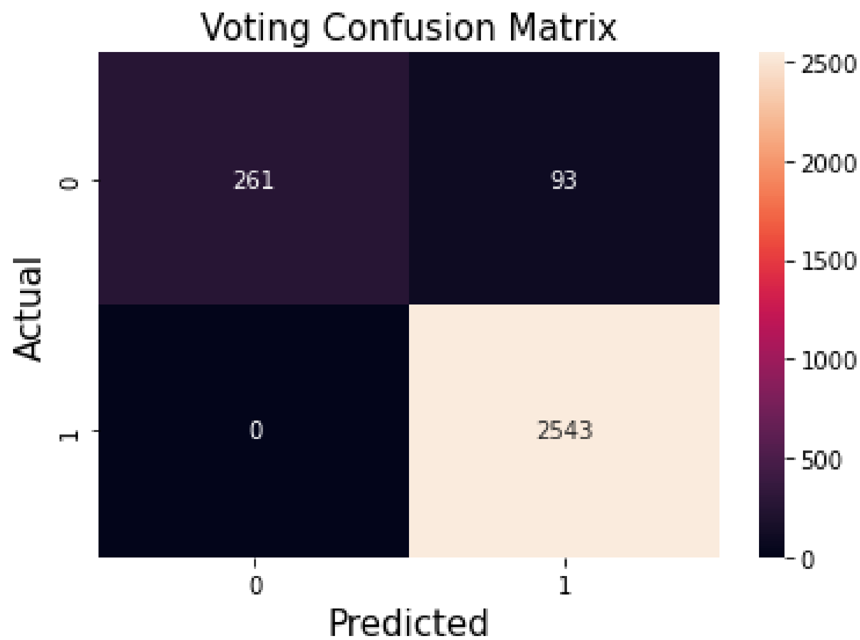
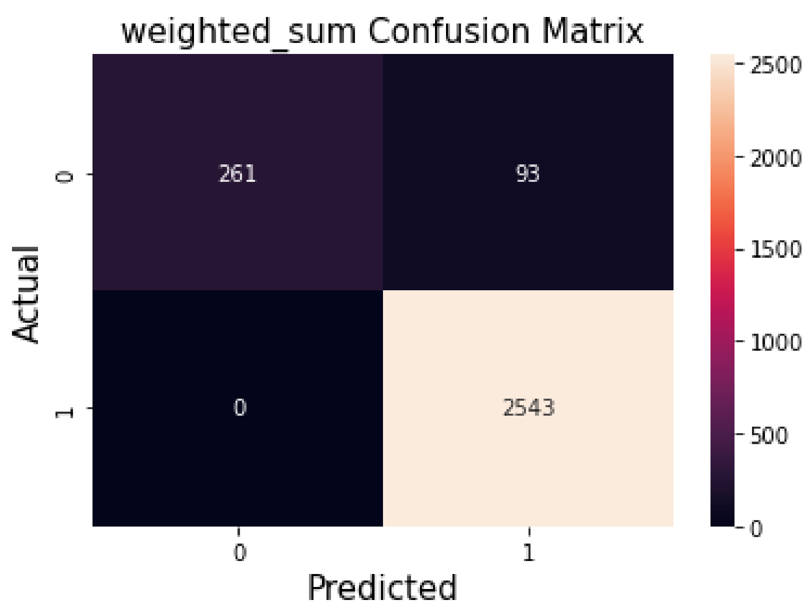
6. Build the weighted sum model manually and plot its confusion matrix

```
#second ensemble framework : weighted sum aggregation
total = accuracy_train_RF + accuracy_train_AB + accuracy_train_NB
w_RF, w_AB, w_NB = accuracy_train_RF/total, accuracy_train_AB/total, accuracy_train_NB/total
aggregated_output = (w_RF * y_test_pred_RF) + (w_AB * y_test_pred_AB) + (w_NB * y_test_pred_NB)

y_pred_weighted_sum  = []
for i in aggregated_output:
    if i > 0.5: y_pred_weighted_sum.append(1)
    else: y_pred_weighted_sum.append(0)

y_pred_weighted_sum = pd.DataFrame(y_pred_weighted_sum).squeeze()
accuracy_weighted_sum = accuracy_score(y_test, y_pred_weighted_sum)
report_weighted_sum = classification_report(y_test, y_pred_weighted_sum)
cf_weighted_sum = ConfusionMatrix(y_test, y_pred_weighted_sum)
PLOT_ConfusionMatrix(cf_weighted_sum,'weighted_sum Confusion Matrix')
```

<table>
<tr>
<td>Confusion matrix of the first 3 models</td>
<td>


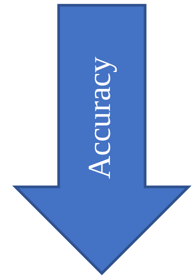


</td>
</tr>
</table>

	
Confusion matrix of Aggregator Models	 

Comparison

Accuracies on test for 5 models



Key	Type	Size	Value
RF_Test	float64	1	0.9982740766309975
Voting_Test	float64	1	0.9678978253365551
weighted_sum_Test	float64	1	0.9678978253365551
AB_Test	float64	1	0.9540904383845358
NB_Test	float64	1	0.8778046254746289

Accuracies on train for the first 3 models

accuracy_train_RF	float64	1	1.0
accuracy_train_NB	float64	1	0.8669198239406231
accuracy_train_AB	float64	1	0.9537412617588676

<table>
<tr><th>Algorithm</th><th>Classification Report</th></tr>
<tr><td>Random Forest</td><td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>0</td><td>1.00</td><td>0.99</td><td>0.99</td><td>354</td></tr>
<tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td><td>2543</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>1.00</td><td>2897</td></tr>
<tr><td>macro avg</td><td>1.00</td><td>0.99</td><td>1.00</td><td>2897</td></tr>
<tr><td>weighted avg</td><td>1.00</td><td>1.00</td><td>1.00</td><td>2897</td></tr>
</table>
</td></tr>
<tr><td>Adaboost</td><td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>0</td><td>0.87</td><td>0.74</td><td>0.80</td><td>354</td></tr>
<tr><td>1</td><td>0.96</td><td>0.98</td><td>0.97</td><td>2543</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.95</td><td>2897</td></tr>
<tr><td>macro avg</td><td>0.92</td><td>0.86</td><td>0.89</td><td>2897</td></tr>
<tr><td>weighted avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>2897</td></tr>
</table>
</td></tr>
<tr><td>Naive Bayes</td><td>
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>0</td><td>0.00</td><td>0.00</td><td>0.00</td><td>354</td></tr>
<tr><td>1</td><td>0.88</td><td>1.00</td><td>0.93</td><td>2543</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.88</td><td>2897</td></tr>
<tr><td>macro avg</td><td>0.44</td><td>0.50</td><td>0.47</td><td>2897</td></tr>
<tr><td>weighted avg</td><td>0.77</td><td>0.88</td><td>0.82</td><td>2897</td></tr>
</table>
</td></tr>
</table>

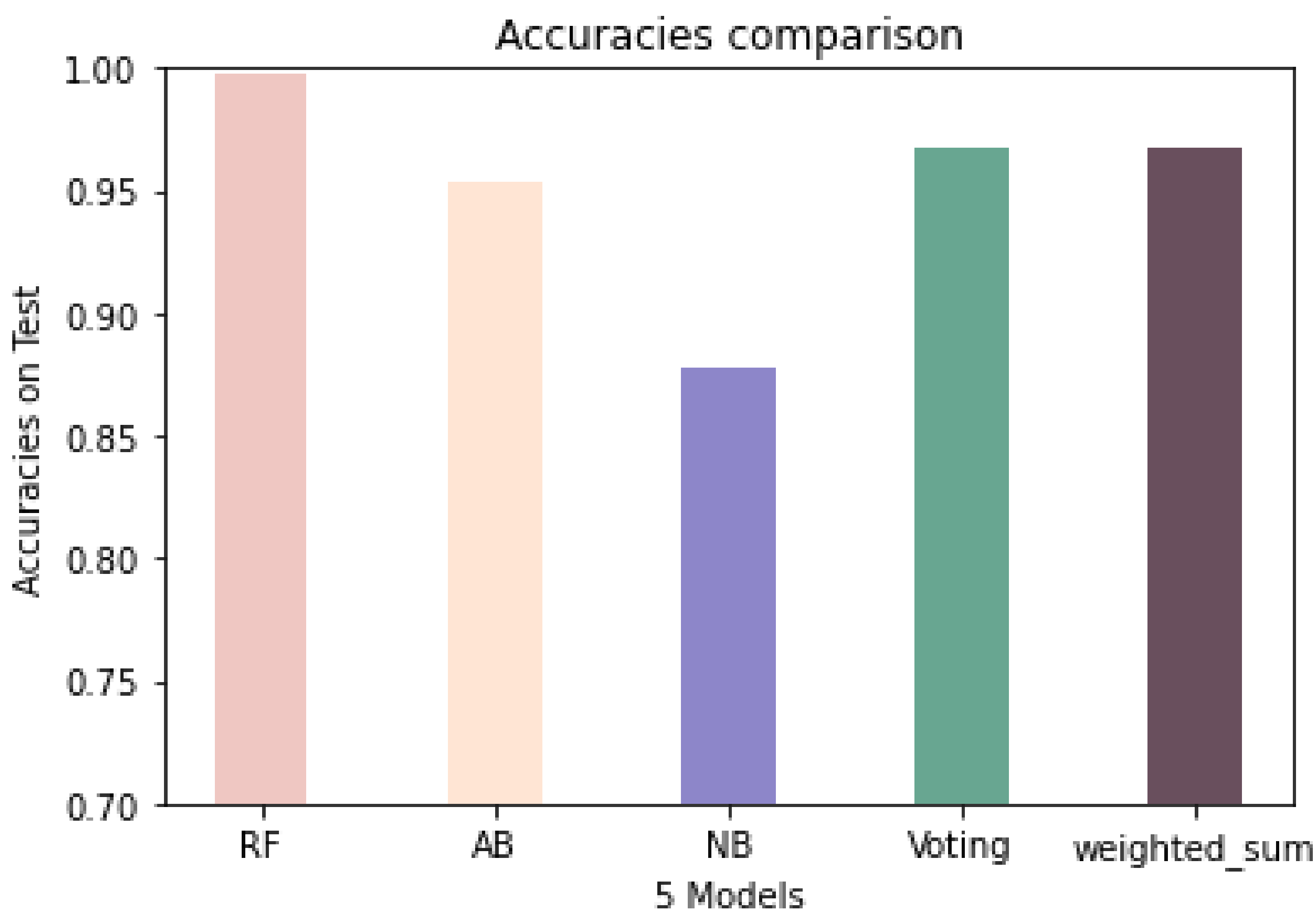
majority voting-based

	precision	recall	f1-score	support
0	1.00	0.74	0.85	354
1	0.96	1.00	0.98	2543
accuracy			0.97	2897
macro avg	0.98	0.87	0.92	2897
weighted avg	0.97	0.97	0.97	2897

weighted sum

	precision	recall	f1-score	support
0	1.00	0.74	0.85	354
1	0.96	1.00	0.98	2543
accuracy			0.97	2897
macro avg	0.98	0.87	0.92	2897
weighted avg	0.97	0.97	0.97	2897

Bar Plots



```
#Bar Plots
Bars = ['RF','AB','NB','Voting', 'weighted_sum']
colors = ['#EFC7C2' , '#FFE5D4','#8D86C9','#68A691','#694F5D']
accuracies = [accuracy_test_RF,accuracy_test_AB,accuracy_test_NB,accuracy_Voting, accuracy_weighted_sum]
plt.bar(Bars, accuracies, 0.4, color = colors )
plt.xlabel("5 Models")
plt.ylabel('Accuracies on Test')
plt.ylim(0.7,1)
plt.title("Accuracies comparison")
```

Conclusion

After we evaluate our models, we noticed that the ‘Random Forest’ model outperformed the other models and even outperformed the ensemble frameworks, and it showed a top-notch accuracy. Also, the two ensemble frameworks showed an excellent performance.