Name : Abdulrahman Almyman ID: [redacted]

IDE: Jupyter Notebook because easy to use and you can divide you work into cells and review your mistakes fast and solving problems. (Note: sometime I use Colab when I am far from my PC)

Lib: pandas (To Preparation dataset easily) , scikit-learn (train_test_split, accuracy_score, RandomForestClassifier, etc.)

1.1.4 :

Chosen Algorithm: Decision Tree Classifier because it is the highest Accuracy

Hyperparameters: I made it default (RandomForestClassifier())

1.1.5.:

-read xAPI-Edu-Data.csv by pandas.read_csv

1- Encoded categorical variables using LabelEncoder

2- Separate features (X) and target (y)

3- Split the dataset into training and testing sets (e.g., 80% train, 20% test) with a random state set to 92 .

4- - using accuracy_score and classification_report to measure its performance on the test dataset also use ConfusionMatrixDisplay.

1.1.6.:

1- IMPORT DATA FROM KAGGEL

2- Data Preparation

3- Encode categorical features

4- # Separate features (X) and target (y)

5- # Split the data into training set (80%) and test set (20%) & RS =91 as the report says

6- # Initialize the Algorithm (chosen: the Random Forest Classifier)

7- # Train the model

8- # Make predictions

9 - # Calculate the accuracy,F1 score, and confusion matrix of the model

1.1.7:

Random Forest Classifier : Model accuracy: 83.33% , F1-score: 0.84

Decision Tree Classifier: Model accuracy: 77.08% , F1-score: 0.78

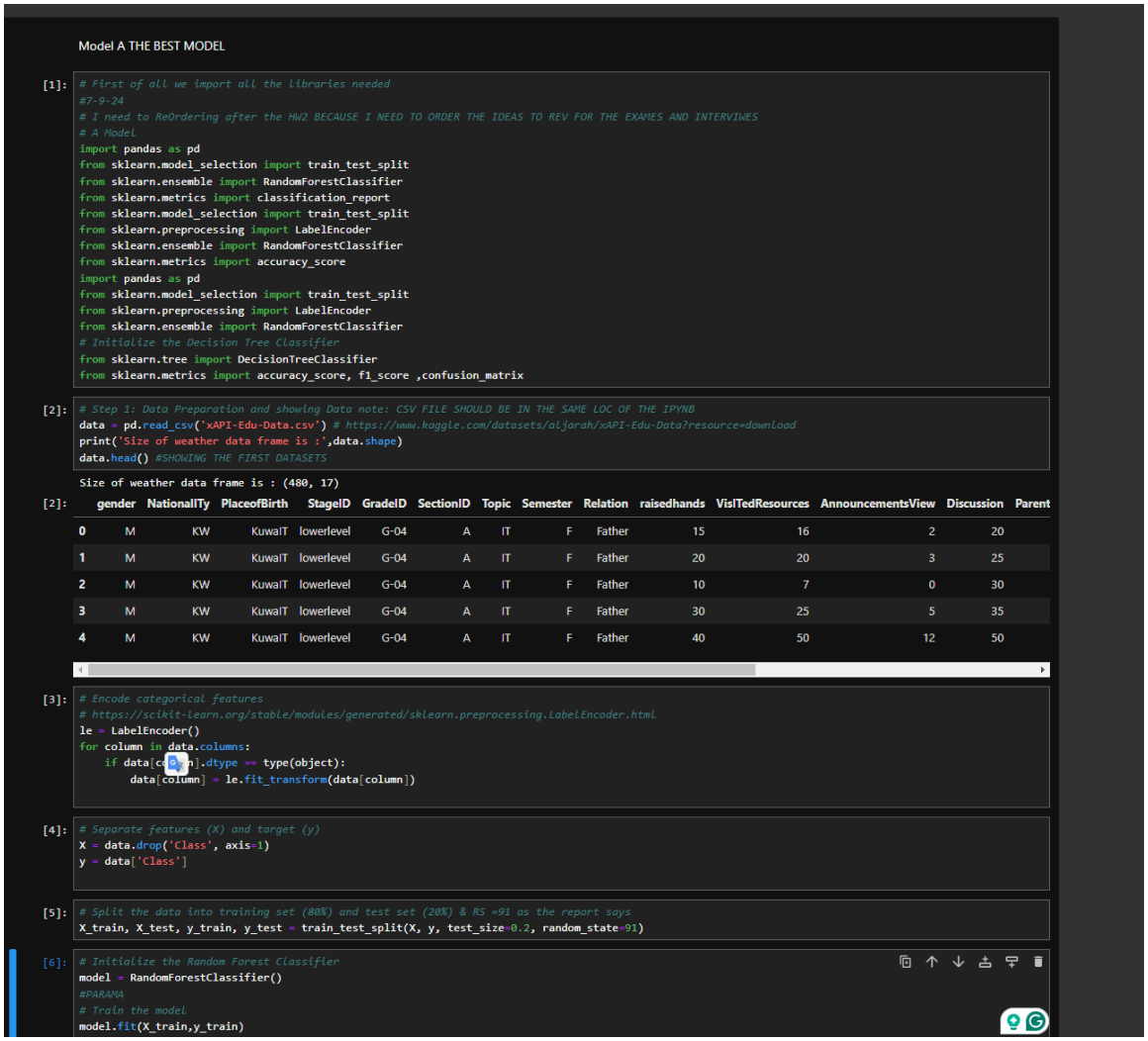
KNN Classifier : Model accuracy: 56.25% , F1-score: 0.57

SVC: Model accuracy: 62.50%, F1-score: 0.63

Logistic : 76.04%, F1-score: 0.77

Last update today HW2_1.1

Photo:



Model A THE BEST MODEL

[1]:
```
# First of all we import all the libraries needed
#7-9-24
# I need to ReOrdering after the HW2 BECAUSE I NEED TO ORDER THE IDEAS TO REV FOR THE EXAMES AND INTERVIWES
# A Model
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
# Initialize the Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, f1_score ,confusion_matrix
```

[2]:
```
# Step 1: Data Preparation and showing Data note: CSV FILE SHOULD BE IN THE SAME LOC OF THE IPYNB
data = pd.read_csv('xAPI-Edu-Data.csv') # https://www.kaggle.com/datasets/aljarah/xAPI-Edu-Data?resource=download
print('Size of weather data frame is :',data.shape)
data.head() #SHOWING THE FIRST DATASETS
```

```
Size of weather data frame is : (480, 17)
```

[2]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion	Parent
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16	2	20	
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20	3	25	
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7	0	30	
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25	5	35	
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50	12	50	

[3]:
```
# Encode categorical features
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
le = LabelEncoder()
for column in data.columns:
    if data[column].dtype == type(object):
        data[column] = le.fit_transform(data[column])
```

[4]:
```
# Separate features (X) and target (y)
X = data.drop('Class', axis=1)
y = data['Class']
```

[5]:
```
# Split the data into training set (80%) and test set (20%) & RS =91 as the report says
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=91)
```

[6]:
```
# Initialize the Random Forest Classifier
model = RandomForestClassifier()
#PARAMA
# Train the model
model.fit(X_train,y_train)
```

[3]:
```
# Encode categorical features
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
le = LabelEncoder()
for column in data.columns:
    if data[column].dtype == type(object):
        data[column] = le.fit_transform(data[column])
```

[4]:
```
# Separate features (X) and target (y)
X = data.drop('Class', axis=1)
y = data['Class']
```

[5]:
```
# Split the data into training set (80%) and test set (20%) & RS =91 as the report says
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=91)
```

[6]:
```
# Initialize the Random Forest Classifier
model = RandomForestClassifier()
#PARAMA
# Train the model
model.fit(X_train,y_train)
```

[6]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

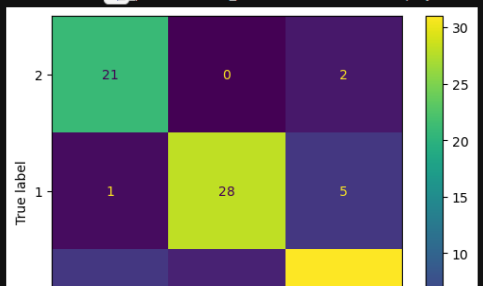
[7]:
```
# Make predictions
y_pred = model.predict(X_test)
```

[8]:
```
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
```

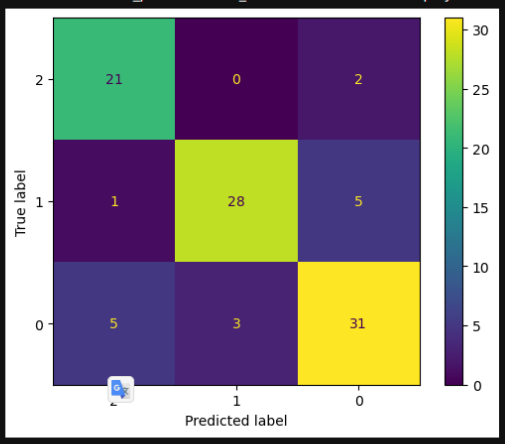
[9]:
```
# Calculate the F1-score of the model
f1 = f1_score(y_test, y_pred, average='macro')
```

[10]:
```
# Calculate the confusion matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
target_names = y.unique()
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=target_names )
disp.plot()
```

[10]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x201e1894bc0>
```



```
[7]: # Make predictions
     y_pred = model.predict(X_test)

[8]: # Calculate the accuracy of the model
     accuracy = accuracy_score(y_test, y_pred)

[9]: # Calculate the F1-score of the model
     f1 = f1_score(y_test, y_pred, average='macro')

[10]: # Calculate the confusion matrix
      from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
      target_names = y.unique()
      cm = confusion_matrix(y_test, y_pred)
      disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=target_names )
      disp.plot()

[10]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x201e1894bc0>
```



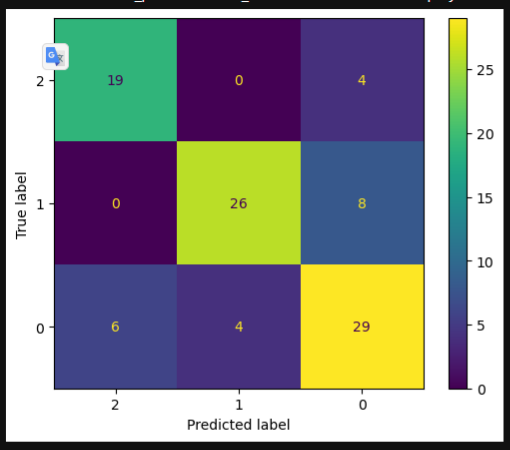

```
[11]: print(f"Model accuracy: {accuracy * 100:.2f}%")
      print(f"F1-score: {f1:.2f}")
      print("Confusion Matrix:\n", cm)

      Model accuracy: 83.33%
      F1-score: 0.84
      Confusion Matrix:
       [[21  0  2]
       [ 1 28  5]
       [ 5  3 31]]
```

Model B

The other results

DecisionTreeClassifier

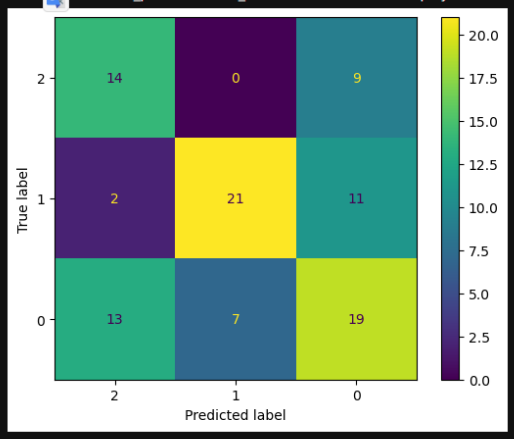
```
[34]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x201de679010>
```




```
[35]: print(f"Model accuracy: {accuracy * 100:.2f}%")
      print(f"F1-score: {f1:.2f}")
      print("Confusion Matrix:\n", cm)

Model accuracy: 77.08%
F1-score: 0.78
Confusion Matrix:
 [[19  0  4]
 [ 0 26  8]
 [ 6  4 29]]
```

KNeighborsClassifier

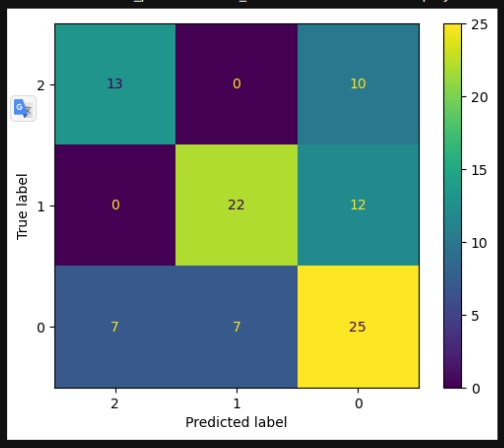
```
[46]: <skl.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x201e38aa600>
```




```
[47]: print(f"Model accuracy: {accuracy * 100:.2f}%")
      print(f"F1-score: {f1:.2f}")
      print("Confusion Matrix:\n", cm)

Model accuracy: 56.25%
F1-score: 0.57
Confusion Matrix:
 [[14  0  9]
 [ 2 21 11]
 [13  7 19]]
```

SVC

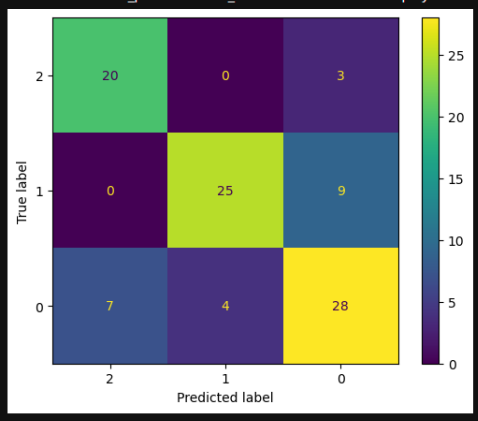
```
[58]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x201e386f3e0>
```




```
[59]: print(f"Model accuracy: {accuracy * 100:.2f}%")
      print(f"F1-score: {f1:.2f}")
      print("Confusion Matrix:\n", cm)

      Model accuracy: 62.50%
      F1-score: 0.63
      Confusion Matrix:
       [[13  0 10]
       [ 0 22 12]
       [ 7  7 25]]
```

LogisticRegression

```
[70]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x201e38b36b0>
```




```
[71]: print(f"Model accuracy: {accuracy * 100:.2f}%")
      print(f"F1-score: {f1:.2f}")
      print("Confusion Matrix:\n", cm)

      Model accuracy: 76.04%
      F1-score: 0.77
      Confusion Matrix:
       [[20  0  3]
       [ 0 25  9]
       [ 7  4 28]]
```

Recources:

https://aws.amazon.com/what-is/hyperparameter-tuning/?nc1=h_ls

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html