MEETING 2

Develop an end to end Machine Learning pipeline

Groupe 6

Content

- Project description
- Wrap-up of meeting's 1

- Meeting 2 agenda
  - Data exploration
  - Data manipulation
  - To do list

Develop an end to end Machine Learning pipeline.

- Project: English quality prediction

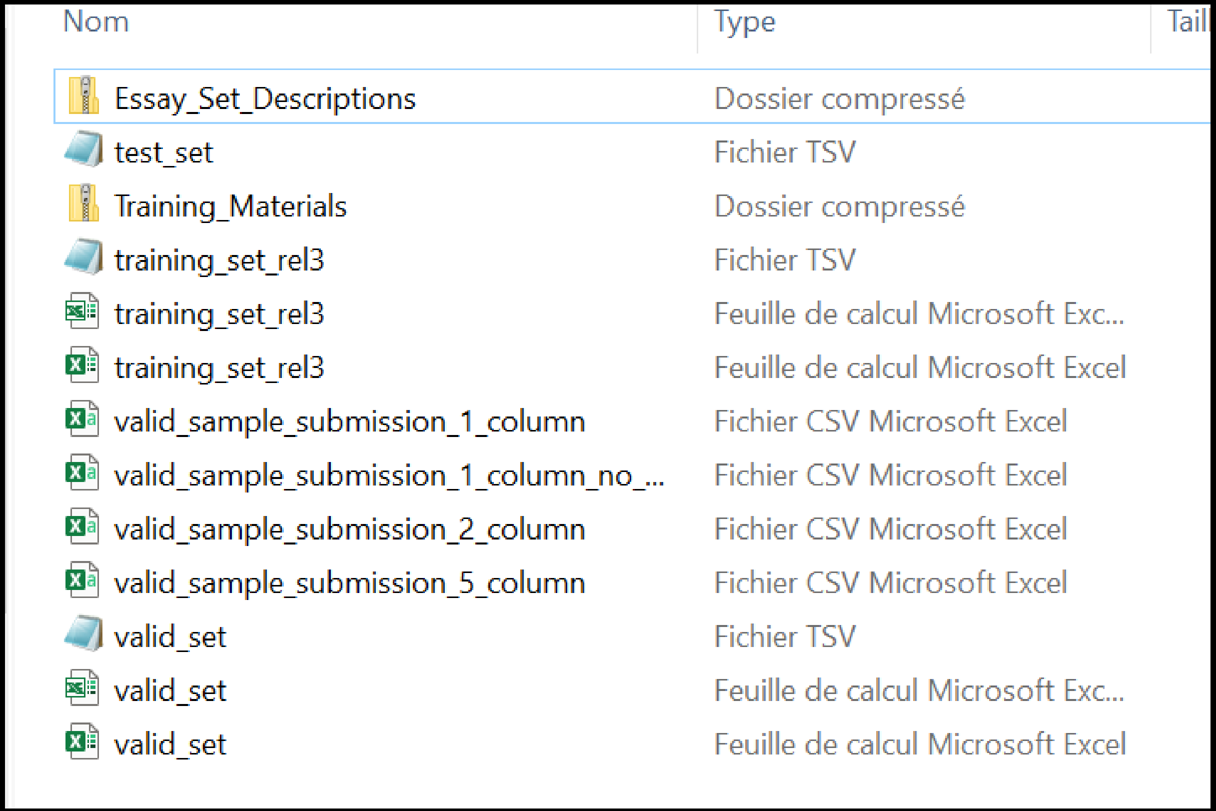
- Develop an end-to-end pipeline that processes an essay and produces a score describing the level of proficiency in English, which we can classify as poor, average or great.

Develop an end to end Machine Learning pipeline.

- This project is a NLP problem that will be the foundation of an English program used by the company Easy Sailing Language Training. Their ambition is to have a reliable tool to assess new students' ability to write in English according to the IELTS grading system. In turn it would help prospective students in knowing how much time they need to invest to get to the next level.
- The goal is to create a reliable tool to assess new students' ability to write in English according to the IELTS grading system.
- In the first time we'll set a simple machine learning model, and in the second time we'll use NLP tools to set a machine learning model.

Project ressources :

- Training set
- Valid set and valid sample
- Test set
- Methodology for dataset creation
- Dataset description



Nom	Type	Taill
Essay_Set_Descriptions	Dossier compressé	
test_set	Fichier TSV	
Training_Materials	Dossier compressé	
training_set_rel3	Fichier TSV	
training_set_rel3	Feuille de calcul Microsoft Exc...	
training_set_rel3	Feuille de calcul Microsoft Excel	
valid_sample_submission_1_column	Fichier CSV Microsoft Excel	
valid_sample_submission_1_column_no_...	Fichier CSV Microsoft Excel	
valid_sample_submission_2_column	Fichier CSV Microsoft Excel	
valid_sample_submission_5_column	Fichier CSV Microsoft Excel	
valid_set	Fichier TSV	
valid_set	Feuille de calcul Microsoft Exc...	
valid_set	Feuille de calcul Microsoft Excel	

- **Dataset:** https://www.transferxl.com/download/08vSFcz3B7fXPr
- **Links:**

  https://paperswithcode.com/

  https://www.analyticsvidhya.com/blog/2021/04/a-guide-to-feature-engineering-innlp/

  https://spacy.io/usage/spacy-101

Wrap-up of meeting 1

- The process must be able to understang the quality of essais
- The specials characters like @ORGANISATION…@CAPS 1 permit the anonymization of personnals informations to prevent the identification of individuals.
- Advices and tips
- How to measure the quality of a student evaluation ?
- The number word
- The the length of a word
- The number of paragraph
- The orthograph and the grammar

Librairies & data uploading

Librairies

- pandas
- scikit-learn
- Spacy
- nltk
- openpyxl
- textstat
- autocorrect
- lexicalrichness
- Seaborn
- matplotlib.pyplot
- readability
- Sklearn
- re

Data_set

- Valid_set
- Training_set
- Sample_score
- Test_set

Data exploration

Go to the content

Data exploration

Entrée [9]:
```
# import training set
training_set= pd.read_excel('C:/Users/User/Desktop/DSTI/Project 2/asap-aes/training_set_rel3.xlsx')
```

Entrée [10]:
```
training_set.describe()
```

Out[10]:

	essay_id	essay_set	rater1_domain1	rater2_domain1	rater3_domain1	domain1_score	rater1_domain2	rater2_domain2	domain2_score	rater1_tra
count	12978.000000	12978.000000	12977.000000	12977.000000	128.000000	12977.000000	1800.000000	1800.000000	1800.000000	2292.0000
mean	10295.432809	4.179458	4.126840	4.137089	37.828125	6.799723	3.333889	3.330556	3.333889	2.4441
std	6308.588616	2.136749	4.212537	4.264320	5.240829	8.970558	0.729103	0.726807	0.729103	1.2117
min	1.000000	1.000000	0.000000	0.000000	20.000000	0.000000	1.000000	1.000000	1.000000	0.0000
25%	4439.250000	2.000000	2.000000	2.000000	36.000000	2.000000	3.000000	3.000000	3.000000	2.0000
50%	10045.500000	4.000000	3.000000	3.000000	40.000000	3.000000	3.000000	3.000000	3.000000	2.0000
75%	15680.750000	6.000000	4.000000	4.000000	40.000000	8.000000	4.000000	4.000000	4.000000	3.0000
max	21633.000000	8.000000	30.000000	30.000000	50.000000	60.000000	4.000000	4.000000	4.000000	6.0000

8 rows × 27 columns

- 12978 id_essay
- 12977 domain1_score (missing values)
- 1800 domain2_score

Data exploration

Entrée [18]:
```
valid_set.describe()
```

Out[18]:

	essay_id	essay_set	domain1_predictionid	domain2_predictionid
count	4218.000000	4218.000000	4218.000000	600.000000
mean	11282.446420	4.123518	13735.433618	7178.000000
std	6173.633131	2.117188	6964.020021	346.698716
min	1788.000000	1.000000	1788.000000	6579.000000
25%	5243.250000	2.000000	7508.500000	6878.500000
50%	10995.500000	4.000000	13995.500000	7178.000000
75%	16852.750000	6.000000	19852.750000	7477.500000
max	21938.000000	8.000000	24938.000000	7777.000000

- 4218 essay_id
- 4218 essay_set
- 4218 domain1_predictionid
- 600 domain2_predictionid
- The scores are missing

Data exploration

Entrée [20]:
```
sample_score.describe()
```

Out[20]:

	prediction_id	essay_id	essay_set	essay_weight	predicted_score
count	4818.000000	4818.000000	4818.00000	4818.000000	4818.000000
mean	12918.816729	10509.725820	3.85907	0.875467	6.240764
std	6867.367811	6129.318271	2.10140	0.216259	8.308969
min	1788.000000	1788.000000	1.00000	0.500000	0.000000
25%	7193.250000	5085.250000	2.00000	1.000000	2.000000
50%	13694.500000	10694.500000	4.00000	1.000000	3.000000
75%	19702.750000	16702.750000	6.00000	1.000000	7.000000
max	24938.000000	21938.000000	8.00000	1.000000	50.000000

- 4818 prediction_id
- 4818 essay_id
- 4818 essay_set
- 4818 essay_weight
- 4818 predicted_score

* The predicted_score is the score of the valid_set

Data manipulation

Go to the content

Data cleanning

Entrée [41]:
```
# Replace empty cells with the corresponding mean
training_set1['score']=training_set1.apply(lambda row: mean_score[row['essay_set']]
```

Entrée [42]:
```
training_set1.isnull().sum()
```

Out[42]:
```
essay_id              0
essay_set             0
essay                 0
rater1_domain1        1
rater2_domain1        1
rater3_domain1    12850
score                 0
rater1_trait1     10686
rater1_trait2     10686
rater1_trait3     10686
rater1_trait4     10686
rater1_trait5     12255
rater1_trait6     12255
rater2_trait1     10686
rater2_trait2     10686
rater2_trait3     10686
rater2_trait4     10686
rater2_trait5     12255
rater2_trait6     12255
rater3_trait1     12850
rater3_trait2     12850
rater3_trait3     12850
rater3_trait4     12850
rater3_trait5     12850
rater3_trait6     12850
level                 0
dtype: int64
```

- We've computed the average of above score for each essay_set and replaced the missing value.
- We've removed all computed data, because we think that they are useless for our training model

Data cleanning

```
Entrée [24]: # merge validset1 and score
valid_score1= pd.merge(valid_set1,sample_score[['prediction_id','predicted_score']], left_on='domain1_predictionid', right_on='pr
```

```
Entrée [25]: print(valid_score1)
```

```
      essay_id  essay_set                                              essay  \
0         1788          1  Dear @ORGANIZATION1, @CAPS1 more and more peop...
1         1789          1  Dear @LOCATION1 Time @CAPS1 me tell you what I...
2         1790          1  Dear Local newspaper, Have you been spending a...
3         1791          1  Dear Readers, @CAPS1 you imagine how life woul...
4         1792          1  Dear newspaper, I strongly believe that comput...
...        ...        ...                                                ...
4213     21933          8   Have you ever noticed that if two little kids...
4214     21934          8                               Laughter @CAPS1 I ...
4215     21935          8   Laughter in @CAPS1 A laugh is not just an act...
4216     21937          8    LAUGHTER @CAPS1 i was younger my friend live...
4217     21938          8   You know how the saying goes live, laugh, lov...

      domain1_predictionid  prediction_id  predicted_score
0                     1788           1788                7
1                     1789           1789                8
2                     1790           1790                9
3                     1791           1791                9
4                     1792           1792                9
...                    ...            ...              ...
4213                 24933          24933               33
4214                 24934          24934               35
4215                 24935          24935               38
4216                 24937          24937               32
4217                 24938          24938               39
```

Merge valid_set and sample_score to have the score in valid set

Classification of score

- This a function to assign a English level to each score for a three-level classification: poor, average and great.

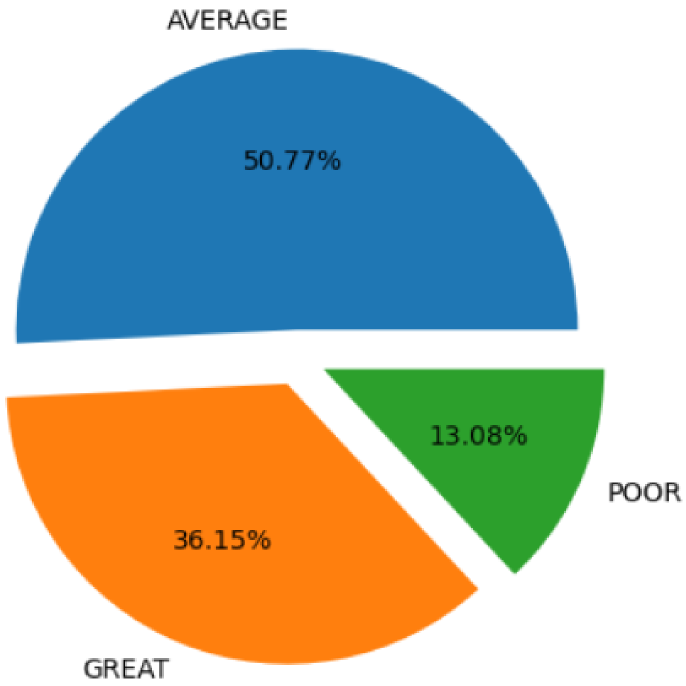
Entrée [33]:

```python
#Define a function to assign a English level to each score
def assign_level (score, essay_set):
    if essay_set == 1:
         poor_limit, average_limit = 6, 10
    elif essay_set == 2:
         poor_limit, average_limit = 3, 5
    elif essay_set == 3:
         poor_limit, average_limit = 1, 2
    elif essay_set == 4:
        poor_limit, average_limit = 1, 2
    elif essay_set == 5:
        poor_limit, average_limit = 2, 3
    elif essay_set == 6:
        poor_limit, average_limit = 2, 3
    elif essay_set == 7:
        poor_limit, average_limit = 15, 23
    elif essay_set == 8:
        poor_limit, average_limit = 30, 45
    else:
        raise ValueError(f"Unknown essay_set {essay_set}")

    if score < poor_limit:
        return 'POOR'
    elif score < average_limit:
        return 'AVERAGE'
    else:
        return 'GREAT'
def classify_levels(dataframe, essay_set='essay_set', score='score', level='level'):
    if level not in dataframe.columns:
        dataframe[level]= None

    dataframe[level] = dataframe.apply(lambda row: assign_level(row[score], row[essay_set]), axis=1)
```

Classification of score

```
Entrée [42]: #try to plot the distridution of differents level on the training set

             # Calculate the explode parameter based on the number of levels
             explode = [0.1] * len(training_set1.level.unique())
```

```
Entrée [43]: #let's plot and display the distribution
             plt.pie(training_set1.level.value_counts(),labels=training_set1.level.value_counts().index,autopct='%1.2f%%',explode=explode)
             plt.show()
```



Split dataset

- First with respect of domain_score 1

  (training_set1, valid_set1, valid_score1)

- Second with respect of domain_score2

  (training_set2 , valid_set2, valid_score2)

Prediction model

```python
def custom_label_encoding(dataframe, column_name, order):
    # Convert the 'level' column to categorical
    dataframe[column_name] = pd.Categorical(dataframe[column_name], categories=order, ordered=True)
    # Create a LabelEncoder
    label_encoder = LabelEncoder()
    # Apply label encoding to the specified column
    dataframe[column_name + '_encoded'] = label_encoder.fit_transform(dataframe[column_name])
```

```python
# first model (set1)
```

```python
print(training_set1.dtypes)
```

```
essay_id       int64
essay_set      int64
essay         object
score        float64
level         object
dtype: object
```

```python
custom_label_encoding(training_set1,'level',order=['POOR', 'AVERAGE', 'GREAT'])
custom_label_encoding(valid_score1,'level',order=['POOR', 'AVERAGE', 'GREAT'])
```

- we are going to transform categorical data (level) to numbers
- let's assigns a unique numerical label to each category
- Encoding of dataset

```python
print(training_set1.dtypes)
```

```
essay_id            int64
essay_set           int64
essay              object
score             float64
level            category
level_encoded       int32
dtype: object
```

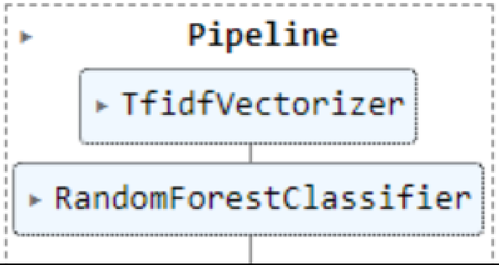
Training of Machine Learning model

```
# Try an ensemble classifier: Random Forest
from sklearn.ensemble import RandomForestClassifier
```

```
# Train-Test Split
x_train=training_set1 ['essay']
y_train=training_set1 ['level_encoded']
x_test=valid_score1 ['essay']
y_test= valid_score1 ['level_encoded']
```

```
# Create an ML model with Random Forest Classifier
rf_model = Pipeline([('tfidf', TfidfVectorizer()),('rf', RandomForestClassifier(n_estimators=100, random_state=42))])
```

```
#Train the model
rf_model.fit(x_train, y_train)
```



Evaluation of prediction model

```
# Evaluate the model
y_test_pred = rf_model.predict(x_test)
```

```
report_rf = classification_report(y_pred=y_test_pred,y_true=y_test)
```

```
print("Accuracy:", accuracy_score(y_test, y_test_pred))
print("\nReport of Random Forest model:\n", classification_report(y_test, y_test_pred))
```

```
Accuracy: 0.793266951161688

Report of Random Forest model:
               precision    recall  f1-score   support

           0       0.77      0.89      0.83      2270
           1       0.84      0.78      0.81      1552
           2       0.76      0.28      0.41       396

    accuracy                           0.79      4218
   macro avg       0.79      0.65      0.68      4218
weighted avg       0.80      0.79      0.78      4218
```

To do list

Go to the content

To do list

Build a NLP prediction model

1. Create some features to measure text complexity

Many measures exist to compute text complexity based on **words** and **word structure:**

- Flesch reading ease
- Gunning Fog
- Automated Readibility index (ARI)
- Smog Index
- Flesch-Kincaid
- Coleman-Liau
- Dale-Chall Readability

Complexity can also be measured **lexically**:

- Words sophistication thank to a corpus (AWL)
- Words frequency (tf-idf)
- Lexical diversity
- Lexical variation features (ratio of words tagged as adjectives, nouns or verbs) over total number of words.

Complexity can also be measured via the **syntactic structure** of the sentences:

- Roots of Sentence tree
- Length of Sentence tree
- Average number of Connections of Sentence tree at the root level
- Length of clauses

The **Quality** of a text should also include:

- misspelling score
- slur usage
- overusage of punctuation

2. Build a correlationion matrix
3. Use a machine learning model depending on colinear correlation matrix to predicte the level of english
4. Evaluate a model
5. hosting and deployment