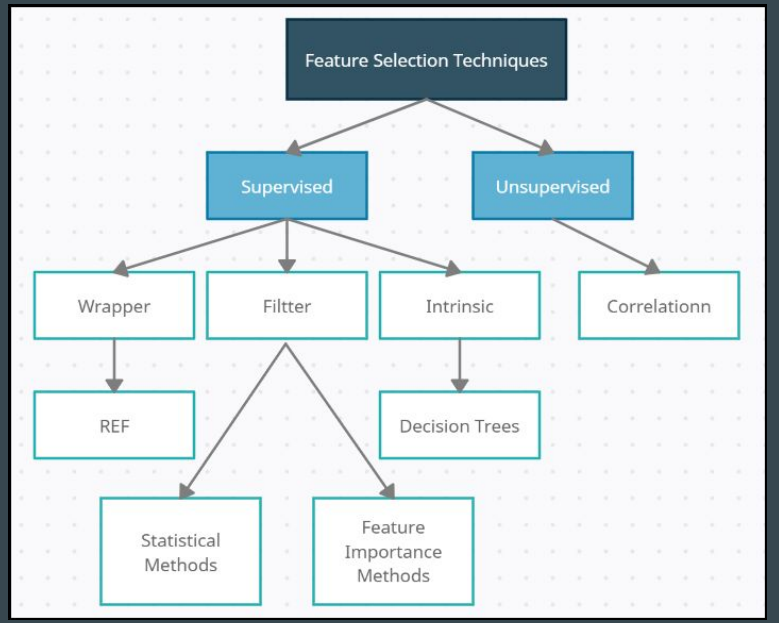
Sentiment Analysis of Video Game Reviews

SCDF
Dennis Rodriguez (0460867), Sergio Soto (0375494)
Francisco Gonzalez (20388005), Christian Narcia (20324290)

1. Feature selection

- The feature selection is the process of selecting the most significant and relevant features from a vast set of features in the given dataset.
- Because our model intends to classify text, our features are coming from words and this represent higher dimensional features.
- We use feature extraction with TF-IDF (Term Frequency-Inverse Document Frequency) to assign a value of how important a word is in each review.



TF-IDF

The TF-IDF uses two statistical methods:

1) The Term Frequency refers to the total number of times a term appears in the document against the total number of words.
2) The Inverse Document Frequency measures how much information the word provides.

The TF-IDF converts raw strings or dataset into vectors. Meaning each word has its own vector and is assigned a TF-IDF score in the process called *vectorization.*

Then, the algorithm compares the similar reviews by looking at the TF-IDF scores of each vector using a Cosine Similarity measurement..

Pros

-Computationally cheap

-Simple implementation using sklearn library function: *TfidfVectorizer.*

Cons

-Cannot carry semantic meaning.

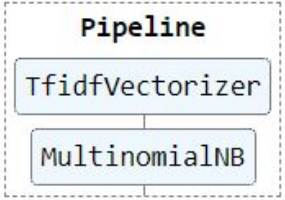
TfidfVectorizer lets you include a list of stop words by setting its parameter stop_words = ‘english’

`sklearn.feature_extraction.text.TfidfVectorizer`

Pipeline

- Here we make a pipeline
  - TfidfVectorizer
  - Multinomial Naive Bayes
- With a pipeline, we must sequentially apply a list of transforms and a final estimator. Intermediate steps of the pipeline must be ‘transforms’, that is, they must implement fit and transform methods. The final estimator only needs to implement fit.

```
set_config(display="diagram")
model
```




```
# Creating a model based on Multinomial Naive Bayes
model = make_pipeline(TfidfVectorizer(min_df=1,stop_words = 'english',),MultinomialNB(alpha = 0.5, class_prior=None, fit_prior = None))
```

Pipeline

- The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. For this, it enables setting parameters of the various steps using their names and the parameter name.
- The 'make_pipeline' function that we use is a utility function that is a shorthand for constructing pipelines.
    - Making pipelines in general becomes easier
- Once the pipeline is made, then you can go ahead and use it like any model, being able to call the '.fit()' function with the training data.

3. Machine Learning model: Naive Bayes

Sklearn Naive Bayes:

Multinomial Naive Bayes

Parameters Used:

Alpha: 0.5

Fit_prior: False

```
MultinomialNB(alpha = 0.5, fit_prior = False)
```

Why Multinomial Naive Bayes?

- Multinomial Naive Bayes is a probabilistic learning method that uses Bayes theorem
- Multinomial Naive Bayes approach is popular in Natural Language Processing
- It calculates the probability of each tag for a given sample
- Using the probability algorithm, it can determine the probability of a review being positive or negative

Bayes theorem: $$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Parameters Used

Alpha

- Smoothing happens in order to avoid 0's in training.
- For example:
  - In a training set:
    - If a positive review contains: "I liked the game unlike **Geometry Dash**"
    - "**Geometry Dash**" is not in the negative set.
  - In testing set:
    - If a negative review contains: "I didn't like **Geometry Dash** either" (It gets marked as a positive review)

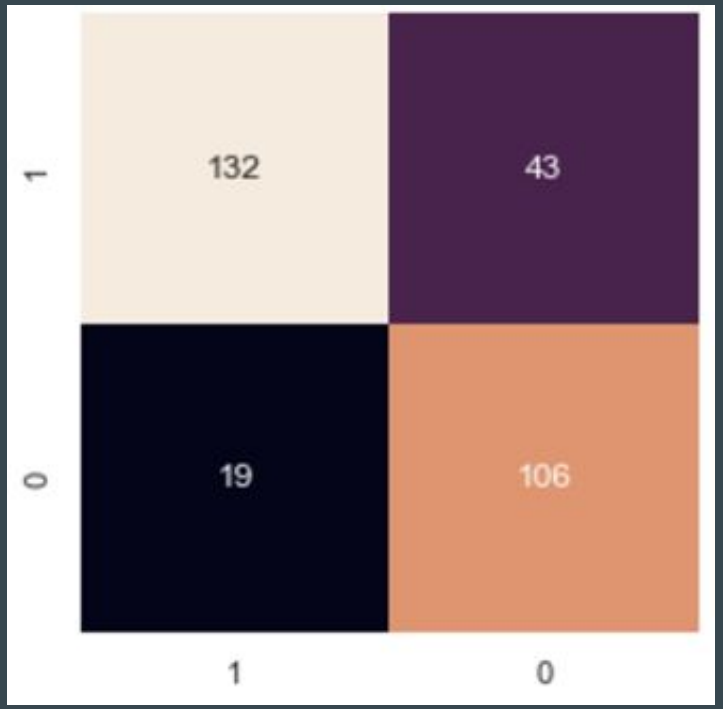
Fit_prior

- Fit_prior was set to False because we wanted the algorithm to use a uniform prior.

Class_prior:

- Default parameter of None.

4. Results and Future Work



Using the trained model to attempt to classify the test data set, we obtain the following confusion matrix (1st attempt):

- 238 reviews correctly classified
- 62 reviews misclassified (acc 0.7933)

```
accuracy_score(y_test, y_pred)
```

```
0.81
```

Future Work - Techniques to Improve Performance

- Modify the parameters of the MultinomialNB function while using different weights in the training step (like 80%/20%)
  - A higher weight on the training set may increase the accuracy of the model, but we want to avoid overfitting
- Use other techniques for text preprocessing such as
  - Stemming
  - Lemmatization
- Use word2vec function instead of TF-IDF