

Exercise 1 : Visualizing Word2Vec Word Embeddings using t-SNE

Word embedding is mapping allowing to represent each word of a given vocabulary by means of a high-dimensional numerical vector (around hundred of components).

The idea is that word embeddings should encode the **semantic** (i.e. the sense of the words). It means that when two words that have close meaning, their representation should be close also : the euclidean distance between their embeddings should be small. For example **mom** and **dad** should be closer than **socket** and **dad**.

To understand more about words embeddings, and see how we can visualize clusters of words how we can visualize word clusters, we shall combine **Word2Vec** representation and **t-SNE**.

To do so we shall use **gensim** Python library and begin to import the following Python librairies

```
import numpy as np
import gensim.downloader
import matplotlib.pyplot as plt
from matplotlib import cm
from numpy import linalg as LA
```

The embeddings can be downloaded using the following command

```
word2vec = gensim.downloader.load('word2vec-google-news-300')
```

1. We first play with word embeddings to understand more

(a) Calculate the word embedding of the word **Paris**

```
emb_paris=word2vec['Paris']
```

What is the shape of this vector?

(b) We now display the 30 most similar words to the word **Paris** using the function **most_similar**. Comment both the result and the format of the output

(c) To understand more how word embeddings encode semantic, we calculate also the word embedding of the two words **London** and **twitter**. Calculate the cosine similarity of (**Paris,London**) and thereafter that of (**Paris,twitter**) where

$$\text{cosine_sim}(\text{Paris}, \text{London}) = \frac{\langle \text{embedding}(\text{Paris}), \text{embedding}(\text{London}) \rangle}{\|\text{embedding}(\text{Paris})\| \cdot \|\text{embedding}(\text{London})\|}$$

and

$$\text{cosine_sim}(\text{Paris}, \text{twitter}) = \frac{\langle \text{embedding}(\text{Paris}), \text{embedding}(\text{twitter}) \rangle}{\|\text{embedding}(\text{Paris})\| \cdot \|\text{embedding}(\text{twitter})\|}$$

(d) Comment

2. We now create synthetic data, naturally associated to clusters.

```
keys = ['Paris', 'Python', 'Sunday', 'Tolstoy', 'Twitter', 'bachelor',
        'delivery', 'election', 'expensive', 'experience', 'financial', 'food',
        'iOS', 'peace', 'release', 'war']

embedding_clusters = []
word_clusters = []
for word in keys:
    embeddings = []
    words = []
    for similar_word, similarity in word2vec.most_similar(word, topn=30):
        words.append(similar_word)
        embeddings.append(word2vec[similar_word])
    embedding_clusters.append(embeddings)
    word_clusters.append(words)

embedding_clusters = np.array(embedding_clusters)
n, m, k = embedding_clusters.shape
```

- (a) Comment the shape of the object `embedding_clusters`? To what are related `n`, `m` and `k`?
 - (b) What is the type of the object `embedding_clusters`? Can be it directly used as an input of the PCA algorithm of scikit-learn?
3. We now use PCA on our synthetic clusters
- (a) Perform PCA on the synthetic dataset `embedding_clusters` *Hint* : you will need to reshape
https://www.w3schools.com/python/numpy_array_reshape.asp
 - (b) Visualize the different clusters related to each `key`
4. Perform t-SNE and and visualize the different clusters related to each `key` in the t-SNE space
5. Compare both

For the two next exercises, one may rely on the notebook `Introduction to Classical ML models`

Exercise 2 : Classification on the Breast Cancer dataset

The Breast Cancer Wisconsin (Diagnostic) Data Set is a classical dataset widely used in classification. The task consists in predicting if a tumor is malignant (M) or benign (B) depending on certain features, that can be downloaded from the UCI Machine Learning repository. More details here

[https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

We want to compare the performance of Logistic Regression, SVM and Random Forest on this classification task. The confusion matrix is the matrix

$$M = \begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$$

where TP : True Positive, TN : True Negative, FP : False Positive, FN : False Negative. We recall the notion of precision and recall

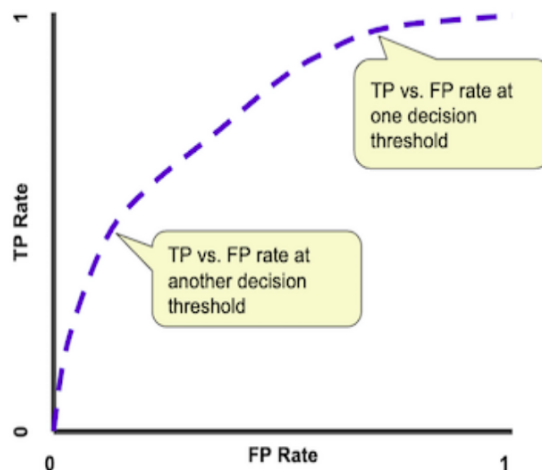
$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN}$$

The accuracy is then defined as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is a very useful metric when all the classes are equally important. But this might not be the case if we are predicting if a patient has cancer. In this example, we can probably tolerate FPs but not FNs. This metric could also be not so relevant when considering imbalanced data.

Most classifiers can be set up so that they output class membership probabilities instead of binary output. In this case, to deduce from this output the class of the observation we need to set a threshold. One can also define the ROC (receiver operating characteristic curve) graph which shows the performance of a classification model at all classification thresholds.



More on ROC curve

<https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

AUC : AUC stands for Area under the ROC Curve. It provides an aggregate measure of performance across all possible classification thresholds.

The higher the area under the ROC curve (AUC), the better the classifier. A perfect classifier would have an AUC of 1. Usually, if your model behaves well, you obtain a good classifier by selecting the value of the threshold that gives TPR close to 1 while keeping FPR near 0.

1. Import the dataset `breast_cancer.csv`. Drop the 'Unnamed: 32' and 'id' columns.
2. In each case, for each classification method (Logistic Regression, SVM and Random Forest)
 - (a) Give the confusion matrix using the function
 - (b) Evaluate the accuracy of the model
 - (c) Plot the ROC curve

Exercise 3 : Deal with imbalanced dataset

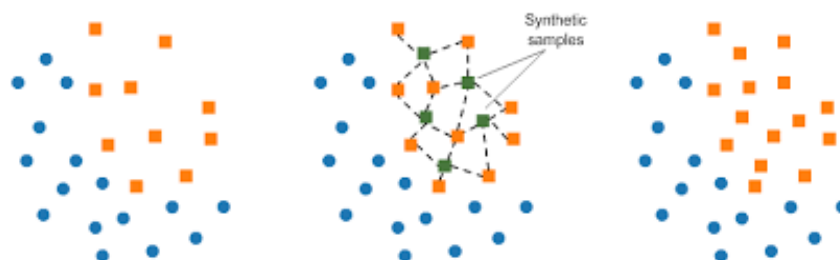
We now consider a dataset about fraudulent credit card transaction

<https://www.kaggle.com/mlg-ulb/creditcardfraud>

1. Import the dataset `creditcardfraud.csv` from the website. What is the distribution of the target variable? Explain the specificity of this dataset.
2. A learning strategy consists in oversampling the minority class using the so-called SMOTE method which is implemented in Python in the library `imblearn`
https://imbalanced-learn.org/stable/generated/imblearn.over_sampling.SMOTE.html
The idea is to synthesize new examples from the minority class. SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors.

The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space.

The synthetic instances are generated as a convex combination of the two chosen instances a and b .



Apply SMOTE to oversample the minority class.

3. For Logistic Regression

- (a) Give the confusion matrix
- (b) Evaluate the accuracy of the model. Do you think this metric is relevant for imbalanced classification?
- (c) An alternative consists in defining Sensitivity-Specificity Metrics. One defines

$$Sensitivity = \frac{TP}{TP + FN}$$

Specificity is the complement to sensitivity, or the true negative rate, and summarises how well the negative class was predicted.

$$Specificity = \frac{TN}{FP + TN}$$

For imbalanced classification, the sensitivity might be more interesting than the specificity. One then defines the G -mean as

$$G - Mean = \sqrt{(Sensitivity \times Specificity)}$$

Use these alternative metrics that can be found at

<https://imbalanced-learn.org/stable/api.html#module-imblearn.metrics>

- (d) We now define the following metrics

$$F_{\beta} = \frac{(1 + \beta^2) \cdot Precision \cdot Recall}{(\beta^2 \cdot Precision + Recall)}$$

When will be the F_1 (resp $F_{0.5}$, F_2 mesure relevant)? What happens in our case?

Hint : use the functions `f1_score` and `fbeta_score` of the library `metrics` of `sklearn`

Exercise 4 : Feature importance and Random Forests

In this exercise we aim at exploring the different concepts of feature importance for Random Forests. To illustrate this question, we consider the following data file `rent.csv` describing New York City apartment rent prices with several additional criteria

1. Download the data `rent.csv`, store it in a dataframe and add the names of the columns to the dataframe. We add also a column of random feature.
2. We explore a first importance measure for Random Forest that is implemented in scikit-learn. Recall that the features for internal nodes are selected with some criterion, Gini impurity for classification tasks, variance reduction for regression. We can measure how each feature decrease the impurity of the split (the feature with highest decrease is selected for internal node). For each feature we can collect how on average it decreases the impurity. The average over all trees in the forest is **the measure of the feature importance** implemented in scikit-learn. See https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html for more details.
 - (a) Train a regressor to predict New York City apartment rent prices using four apartment features. Thereafter calculate the Random Forest Built-in Feature Importance for the data `rent.csv` of each feature using the function `feature_importances_`. Comment
 - (b) Train a classifier predicting apartment interest level (low, medium, high) using 5 features + the random column. Thereafter calculate the Random Forest Built-in Feature Importance for the data `rent.csv` of each feature using the function `feature_importances_`. Comment
3. Breiman defined also another importance measure called **permutation measure** defined in the following way. Record a baseline accuracy (classifier) or R2 score (regressor). Permute the column values of a single predictor feature and then pass all test samples back through the Random Forest and recompute the accuracy or R2. The importance of that feature is the difference between the baseline and the drop in overall accuracy or R2 caused by permuting the column. Calculate it in the two previous cases using the function `permutation_importance` of the library scikit-learn. Comment

Exercise 5 : Quantile regression with ensemble methods

In this exercise, we consider the results of a survey given to visitors of hostels listed on Booking.com and TripAdvisor.com. Our features here are the average ratings for different categories

- "f1": "Staff"
- "f2": "Hostel booking"

- "f3": "Check-in and check-out"
- "f4": "Room condition"
- "f5": "Shared kitchen condition"
- "f6": "Shared space condition"
- "f7": "Extra services"
- "f8": "General conditions conveniences"
- "f9": "Value for money"
- "f10": "Customer Co-creation"

Our target variable is the hostel's overall **rating** on the website. The dataset is `hostel_factors.csv` and can be downloaded on the github repository

1. Fit a Quantile Gradient Boosting Tree using the option `loss='quantile'` in the usual Gradient Boosting Regressor
2. Store the quantiles corresponding to the 97.5th and 2.5th percentile
3. Plot the confidence intervals for the regression