**Binary classification**

The aim of this work is to build a classification model to predict either a candidate is fluent or not, given metrics of phrases types in documents.

For this purpose, we will test 5 classification methods namely: Support vector machines (SVM), Random forests (RF), Naïve Bayes (NB), K nearest neighbors, and neural networks. We will use these methods to build classifiers that can predict the fluency of candidates using a dataset of 464 data point. We will be using Python with scikit-learn machine learning library to preprocess the data then train and fit our model. We also use grid search algorithm to find the optimal parameters of the models and feature selection methods to detect the most important features in our dataset.

1. **Data Preparation:**

We need first to load and clean our data and in order to feed it to our model.

After loading the data set to a pandas dataframe. The variable **ToPredict** is either N or Y. We need to transform it to binary so Y=0 and N=1.

The features:

**Addition\_C, Comparison\_C, Concession\_C, Contrast\_C, Emphasis\_C, Example\_C, Summary\_C, Time sequence\_C, Subject\_P, Object\_P, Possessive\_P, Relative\_P, Demonstrative\_P,**

Are just counts so it would be more accurate to calculate their frequency. We add to the data frame, 13 new columns containing these values divided by the number of words W representing frequency of these features.

In our data set we have some dependencies between the variables, this can affect the accuracy of our model. In order to know what are the variables that we should keep we create new data frames where we drop a set of variables at each time:

**The first data frame** is the original one without dropping any variable.

**In the second data frame,** we keep just the class ‘ToPredict’ and the independent original features: ‘Con', 'Pron','W', and 'S'.

**In the 3rd data frame** we keep just the class ‘ToPredict’ and the independent variables: 'Addition\_C', 'Comparison\_C', 'Concession\_C', 'Contrast\_C', 'Emphasis\_C', 'Example\_C', 'Summary\_C', 'Time sequence\_C', 'Subject\_P', 'Object\_P', 'Possessive\_P', 'Relative\_P', and 'Demonstrative\_P'.

**In the 4th data frame** We keep just the class ‘ToPredict’, “Average S” and the new 13 features representing the frequencies.

We define the input and output vectors X and y. X a matrix with the kept variables for in the data frame and y is the vector ‘ToPredict’.

1. **Split the data:**

We split our data into training data 80% and validation or test data 20%

1. **Classification Models**
2. **Support vector machine****:**

Support vector machine are useful techniques in supervised data classification, they were first invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. The non-linear classification wasn’t suggested until 1992 by [Bernhard E. Boser](https://en.wikipedia.org/w/index.php?title=Bernhard_E._Boser&action=edit&redlink=1), [Isabelle M. Guyon](https://en.wikipedia.org/w/index.php?title=Isabelle_M._Guyon&action=edit&redlink=1) and [Vladimir N. Vapnik](https://en.wikipedia.org/wiki/Vladimir_N._Vapnik). The non-linear classifier was based on kernel trick. The idea of SVM is to find a hyperplane that separates the data set using training data, to classify new data points [1].

**Parameters of the SVM.**

In a SVM there is three main parameters to define:

**C parameter:** the cost of classification. A large C gives low bias and high variance, low bias because we penalize the cost of misclassification a lot. A small C gives higher bias and lower variance [2].

**gamma parameter:** defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’ [2].

**Kernel parameter:** Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, or ‘sigmoid’ [2].

We use the grid search provided by GridSearchCV exhaustively generates candidates from a grid of parameter values specified with the tuned\_parameters.

[{'kernel': ['rbf'], 'gamma': [1e-2, 1e-3, 1e-4], 'C': [1, 10, 100, 1000, 10000]},

{'kernel': ['sigmoid'], 'gamma': [1e-2, 1e-3, 1e-4], 'C': [1, 10, 100, 1000, 10000]},

{'kernel': ['poly'], 'gamma': [1e-2, 1e-3, 1e-4], 'C': [1, 10, 100, 1000, 10000]},

{'kernel': ['linear'], 'C': [1, 10, 100, 1000, 10000]}]

We use two scoring methods to evaluate each set of parameters precision and recall.

We run GridSearchCV and evaluate using each scoring method and print the results.

**Fitting and testing**

We then use the best parameters to fit our model and evaluate accuracy using the test data.

The same procedure is repeated for all the mentioned data frames. The 4th data frame gives the best results. With an accuracy of **0.827956989247** on the test set. The parameters found by the grid search algorithm are: **{'kernel': ‘rbf’, 'C': 100, 'gamma': 0.0001}**.

1. **Random Forests**

Random forest is an ensemble tool which takes a subset of observations and a subset of variables to build a decision trees. It builds multiple such decision tree and amalgamate them together to get a more accurate and stable prediction [3].

**Random forest classifier parameters**

We use Random forest classifier of **Scikit**-**learn** library with the following parameters:

**n\_estimators**: is the number of trees we want to build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance but makes your code slower. We choose a high value because this makes our predictions stronger and more stable. We choose a number of 10000 estimators.

**min\_sample\_leaf**: A smaller leaf makes the model more prone to capturing noise in train data. We choose a number of 50 leafs.

**Feature selection:**

* **Mean decrease impurity feature selection**

The feature selection based on impurity reduction is biased towards preferring variables with more categories when the dataset has two (or more) correlated features, then from the point of view of the model, any of these correlated features can be used as the predictor, with no concrete preference of one over the others. But once one of them is used, the importance of others is significantly reduced since effectively the impurity they can remove is already removed by the first feature. Therefore, they will have a lower reported importance [4].

The results are features sorted by their score of importance as follows:

[(0.2146, 'Average S.'), (0.1343, 'W'), (0.1018, 'Demonstrative\_P/W'), (0.0982, 'Subject\_P/W'), (0.077, 'Pron'), (0.0588, 'Demonstrative\_P'), (0.0533, 'Relative\_P/W'), (0.0326, 'Possessive\_P'), (0.0292, 'Relative\_P'), (0.0251, 'S'), (0.0248, 'Object\_P/W'), (0.0212, 'Possessive\_P/W'), (0.0192, 'Object\_P'), (0.0174, 'Con'), (0.0171, 'Addition\_C/W'), (0.0154, 'Example\_C/W'), (0.0152, 'Contrast\_C/W'), (0.011, 'Addition\_C'), (0.01, 'Time sequence\_C/W'), (0.0057, 'Example\_C'), (0.0051, 'Subject\_P'), (0.0047, 'Time sequence\_C'), (0.0024, 'Contrast\_C'), (0.002, 'Cause\_C/W'), (0.0019, 'Summary\_C/W'), (0.0011, 'Summary\_C'), (0.0007, 'Cause\_C'), (0.0, 'Emphasis\_C/W'), (0.0, 'Emphasis\_C'), (0.0, 'Concession\_C/W'), (0.0, 'Concession\_C'), (0.0, 'Comparison\_C/W'), (0.0, 'Comparison\_C')]

**Fitting and testing**

We then use the best parameters to fit our model and evaluate accuracy using the test data.

The 1st data frame is the one used here because random forests aren’t affected by dependent features. The results give an accuracy of **0.752688172043** on the test set.

1. **Naïve Bayes**

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of independence between every pair of features. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality [5]. We test Naïve Bayes 3 types of distributions namely Bernoulli, Multinomial and gaussian distribution. We will use the fourth data frame as the NB model suppose that the variables are independent.

* **Bernoulli Naïve Bayes:**

This classifier models the likelihood of a feature given a class with a Bernoulli distribution. The results give an accuracy of **0.655913978495** on the test set.

* **Multinomial Naïve Bayes:**

This classifier models the likelihood of a feature given a class with a multinomial distribution. The results give an accuracy of **0.752688172043** on the test set.

* **Gaussian Naïve Bayes**

This classifier models the likelihood of a feature given a class with a Gaussian distribution. The results give an accuracy of **0.731182795699** on the test set.

1. **K nearest neighbors**

K nearest neighbors is one of the most popular classification methods. It is characterized by the simplicity of its model. It consists of calculating the distance between each new point and the closest points then perform a ranking and take the K nearest points. We then take the class of the most frequent class of these points as the class of the new point [6].

**Parameters:**

We define two main parameters namely the number of neighbors and the weighting method. We take a number of 35 neighbors and a weighting method by distance i.e. the closest points have more weight over the far points.

**Results:**

After fitting the model using the 4th, we test the model on test data, we obtain an accuracy of **0.827956989247.**

1. **Artificial neural networks ANN:**

Artificial neural networks are relatively crude electronic networks of neurons based on the neural structure of the brain. They process records one at a time, and learn by comparing their classification of the record (i.e., largely arbitrary) with the known actual classification of the record. The errors from the initial classification of the first record is fed back into the network, and used to modify the networks algorithm for further iterations [7].

We build a multi-layer perceptron with 2 hidden layers. We test several architectures of the network, and the best results was given by 100 neurons in the first hidden layer and 3 neurons in the second layer.

**Parameters:**

The **solver** function for optimization is ‘**lbfgs**’ which is convenient for small data sets, it converges faster and performs better.

**Alpha** is a parameter for regularization term, aka penalty term, that combats overfitting by constraining the size of the weights. Increasing alpha may fix high variance (a sign of overfitting) by encouraging smaller weights, resulting in a decision boundary plot that appears with lesser curvatures. Similarly, decreasing alpha may fix high bias (a sign of underfitting) by encouraging larger weights, potentially resulting in a more complicated decision boundary. We choose a small alpha to avoid overfitting: **alpha = 1e-5**.

**Results:**

After fitting the model using the 4th, we test the model on test data, we obtain an accuracy of **0.817204301075.**

**Summary and discussion:**

The best results were given by the forth data frame, so we considered it for all our analysis. For each classifier, we saved the prediction results of the test data in an excel file. The best predictions were given by SVM, KNN, and ANN, with an accuracy of **0.827956989247** for SVM and KNN, and an accuracy of **0.817204301075** for ANN. The results of all classifiers are given in the following table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SVM** | **Random Forests** | **Naïve Bayes** | | | **KNN** | **ANN** |
| **Bernoulli NB** | **Multinomial NB** | **Gaussian NB** |
| **0.8279569893** | **0.75268817** | **0.6559139785** | **0.75268817** | **0.7311827957** | **0.8279569893** | **0.8172043011** |

These results are pretty good for predicting the fluency of a candidate. We tried to improve the models using different techniques:

* **Features creation:** We created new features to extract accurate information about the text treated. We added 13 new features that gave the best results when handling them in a new data frame.
* **Features selection:** We tested severaldata frames with different features so that we can find the most predicting features in our data set.
* **Parameters tuning:**  We performed grid research algorithms to find the best parameters for our models, we also tested different parameters according to our data set and picked the ones that gave the best results.
* **Different algorithms:** We used here 5 types of classifiers to find out the best model that can classify accurately our candidates.

This Binary classification problem can be improved further by:

* **Using more data:** In machine learning problems, the more data we have the more accurate are the predictions, so if we can get larger data sets we can have better accuracy.
* **New features:** Based on the domain knowledge we can create new features that can help in the model learning. We can also create features by combining two or more existing features (example: relative/demonstrative)
* **Ensemble learning:** Combining different models can give better results, the generalization ability of a combination of learners is usually stronger than that of base learners [8].

**Bibliography**

[1] Cristianini, Nello; and Shawe-Taylor, John; [*An Introduction to Support Vector Machines and other kernel-based learning methods*](http://www.support-vector.net/), Cambridge University Press, 2000. [ISBN 0-521-78019-5](https://en.wikipedia.org/wiki/Special:BookSources/0521780195) (SVM Book).

[2] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A practical guide to support vector classification." (2003): 1-16.

[3] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[4] Louppe, Gilles, et al. "Understanding variable importances in forests of randomized trees." Advances in neural information processing systems. 2013.

[5] Murphy, Kevin P. "Naive bayes classifiers." University of British Columbia (2006).

[6] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879.

[7] <http://www.solver.com/xlminer/help/neural-networks-classification-intro>

[8] Zhi-Hua Zhou, Ensemble Learning, National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China