Baseline code

The iterative development process of the project started with the implementation of the baseline code. In order to keep the code clean and readable, all necessary for its execution libraries were imported at the very beginning. These libraries are:

* Numpy - useful for large, multi-dimensional arrays and matrices; it also contains a vast amount of useful mathematical functions.
* Pandas - it enables a quick and convenient way to load data from a csv file; it provides many Data Frames that allow an easy organization, exploration, representation, and manipulation of the data.
* Matplotlib - it is a comprehensive library for creating static, animated, and interactive visualizations in Python; a number of third-party libraries, such as seaborn can be integrated with it.
* Seaborn - it is a data visualization library based on matplotlib; it provides a high-level interface for drawing attractive and informative statistical graphics.
* Scikit Learn (Sklearn) - it features various classification, regression and clustering algorithms; it interoperates with the Python numerical and scientific libraries. Algorithms that were imported from this package:
  + CountVectorizer - a feature extraction method that converts the text content into numerical feature vectors.
  + TfidfTransformer - transforms a count matrix to a normalized tf or tf-idf representation
  + MultinomialNB - a classifier suitable for classification with discrete features, for example, word counts for text classification.
  + Pipeline - implements utilities to build a composite estimator, as a chain of transforms and estimators
  + train\_and\_test\_split - used for data preparation, by dividing it into training and testing sets.
  + GridSearchCV - searches over specified parameter values for an estimator; uses cross-validation to evaluate each value of a parameter grid and then chooses the best one.
  + Metrics - includes score functions, performance metrics and pairwise metrics, and distance computations.
* Warnings - used to ignore all occurring warnings by using warnings.filterwarnings()

Underneath the imported libraries all functionalities of the program were instantiated.

Firstly, by using the Pandas library the data from a csv file that contains movies plots, genres, titles, etc., was loaded. Subsequently, three functions were called by using Pandas DataFrame. The head() function was used to print the head of a DataFrame in a readable tabular format and display the first ten records of the data used in the program. The shape() function was used to display the dimensionality of the used data. The third function was dtypes(), which clearly depicted the data type of each column from the imported data.

The next step was to split the dataset into training and testing subsets. It is a crucial step in every machine learning process. It was done by using the train\_test\_split() function. The data relevant to our program were genres and plots. Their division proportions were established to 80% of the dataset for training and 20% for testing.

After splitting the data a Pipeline object was created and fitted to the plot and genre data. Methods used in the Pipeline setup were CountVectorizer with the stop-words filtered out in order to gain better performance, TfidfTransformer to normalise the counts to better represent the importance of words and MultinomialNB, which is a common classifier for text classification.

Another action performed by our program was the extraction of the ten most significant features for each movie genre that is in the database. It enabled us to check what words defined particular genres and whether it was rational to take them into account or should they be removed.

Following, a prediction test was carried out on the test set in order to check if the program works properly.

Then, in order to compare previously created pipeline with and without stop-words being filtered out, and with and without using a Tfidf transform, a grid search was used. It clearly estimated which parameters were the best for better performance. Moreover, using the „score” method the accuracy of the model was calculated.

Consecutively, another prediction test was carried out. This time for specific textual data consisting of words that were listed as those that have the greatest significance for a particular movie genre (action). The reason for this was to check whether the program would show the expected movie genre.

Further action in our program was the creation of a diagram with the fifteen most frequently occurring genre types in the chosen dataset. Based on that, we were able to identify to what extent some genres outweigh others, which was very important because such inequalities could also affect the training of the model and cause incorrect prediction of movie genres.

Additionally, in the end, the evaluation of the pipeline performance on the training set was implemented. The confusion matrix was computed to calculate the number of correctly and incorrectly classified examples in the data and based on that numbers a text report was built showing the precision, recall and f1-score metrics.