In this project, we employed feature selection and classification techniques to analyze and model a synthetic dataset. The goal is to have the highest accuracy score, while amplifying the effect of salient features and suppressing the effect of noise and less important features.

For this purpose, we have defined four wrappers to help create a pipeline to load the data from a sql database, create a data dictionary, which will be inputted to a transformer and finally a general model which will use the data dictionary output of the transformer to score the model. These wrapper function - located at the lib folder - are:

- load_data_from_dtabase: connect and download data from a sql database and load the data into a dataframe.
- make_data_dict: splits the dataframe into test and train sets, test size and random state can be modified and returns a data dictionary.
- general_transformer: which uses the data dictionary to apply a transformer which can be modified, and (scales and) transforms the data.
- general_model: finally, the model is inputted through this function and we will have the test/train score and hence the accuracy score.

The data set is called MADELON, an artificial and highly non linear dataset with 2000 instances and and 500 features. Do to the stochastic nature of the dataset we would like to reduce the number of features while obtaining the best accuracy score possible.

We start with a logistic regression with a default penalty of l2. The accuracy score is 0.53 and all the features have an effect on the model and the score. Changing the penalty to l1 actually had a small negative effect on the score but it reduce the number of features to 453. Still high for this dataset.

Next step, we experiment with different inverse of gamma values or C, from 0.001 to 1000 through a grid search along with changing the penalty between l1 and l2. The final result shows significant feature reduction - to two features, 475 and 241 - however the accuracy score is, despite an increase, is still low, at 0.62.

So, at the third and final step we will deploy grid search to apply it to our models. In addition to our logistic classifier we will use the kNN classifier. We will scale the data for both models, and we will use k Best transformer and input the data to logistic regression and kNN classifiers.

To see the best result of the combination of kBest/LogisticRegression and kBest/kNN models we will use grid search to search and find the best parameters for our transformer and classifiers.

The k range is set to 3 to 30. Hopefully the best k is in this range. We change the penalty between l1 and l2 and C from 0.001, 1 and 100. The best result we get is with l2 penalty, a C of 0.0212 and a test score of 0.64. The best model is:

```
LogisticRegression(C=0.021181818181818184, class_weight=None, dual=False,
    fit_intercept=True, intercept_scaling=1, max_iter=100,
    multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
    solver='liblinear', tol=0.0001, verbose=0, warm_start=False))
```

We would like to have a better score. So we try kNN.

At this stage we kNN model along with same k Best transformer. The best result is achieved with k=13, n-neighbors=5 and the test score is 0.88.

So to conclude we achieved our goal, best accuracy score, by utilizing a standard scaler, a kBest transformer and kNN classifier and we improved the score from 0.53 with a default logistic regression to 0.88.