University of Liège

ELEN0062: Introduction to Machine Learning
Project 3

Michele Spano - S194733
Rodolphe Ponthon - S195539
Boris Brogle - S195545

December 2019

Contents

1 Our best working solution

At the end of the challenge, we came up with that solution:

First of all, we changed the feature extraction function in order to increase the number of features extracted. To do so, we added 3 others functions/algorithms to extract features from the SMILES, we indeed added the `MACCSkeys`, the `RDKFingerprint` and the `LayeredFingerprint` functions, bringing us to 4775 features in total (against the 124 at the beginning).

The next step was to select the best features in order to reduce the computation time but also increase the performance of our model. For this purpose, we used `SelectKBest` as features selection model with $k = 1000$.

To train the model we used the test and set method, so we splitted the learning set to have 66% of it as our training set and the remaining 33% as our validation set.

Before training, we changed the 0 to 1 ratio in the training sample (not on the validation sample) so that to fifteen 0s corresponds one 1, while the ratio on the whole learning sample was initially around thirty to one.

Finally we trained the model. The model used is the `RandomForestClassifier` with 8000 estimators. The model has been trained on the training set to predict the auc score on the validation set. Before submitting, we trained the model on the whole learning set in which we discarded half the 0s to get the fifteen to one ratio.

You'll find below the code used to generate or best submission:

```python
    def create_fingerprints(chemical_compounds, fptype="rdkit"):
    n_chem = chemical_compounds.shape[0]

    nBits = [512, 167, 2048, 2048]
    X = np.zeros((n_chem, nBits[0]))
    X2 = np.zeros((n_chem, nBits[1]))
    X3 = np.zeros((n_chem, nBits[2]))
    X4 = np.zeros((n_chem, nBits[3]))
    for i in range(n_chem):
        m = Chem.MolFromSmiles(chemical_compounds[i])
        X[i,:] = AllChem.GetMorganFingerprintAsBitVect(m,3,nBits=nBits[0],
  useFeatures=True)
        X2[i,:] = Chem.MACCSkeys.GenMACCSKeys(m)
        X3[i,:] = Chem.RDKFingerprint(m)
        X4[i,:] = Chem.LayeredFingerprint(m)
    Xret = np.concatenate((X,X2,X3,X4), axis=1)
    return Xret

```

Listing 1: Our feature extraction function

```python
    # Fingerprint creation
    with measure_time("Creating learning sample fingerprint"):
        X_WholeSet = create_fingerprints(LS["SMILES"].values)
    y_WholeSet = LS["ACTIVE"].values

    TS = load_from_csv(args.ts)
    with measure_time("Creating test sample fingerprint"):
        X_TS = create_fingerprints(TS["SMILES"].values)

    # Feature selection
    _k = 1000
    with measure_time("Selecting the "+str(_k)+" best features"):
        featureSelector = SelectKBest(chi2, k=_k)
        X_WholeSet = featureSelector.fit_transform(X_WholeSet, y_WholeSet)
        X_TS = featureSelector.transform(X_TS)

    # Splitting
    with measure_time("Splitting and shuffling the datas"):
        X_LS, X_VS, y_LS, y_VS = train_test_split(X_WholeSet, y_WholeSet,
test_size=0.33, random_state=42)


    WS = list(zip(X_WholeSet, y_WholeSet))
    LS = list(zip(X_LS, y_LS))
    VS = list(zip(X_VS, y_VS))

    # Changing 0 to 1 ratio in learning samples
    ratio = 15
    LS0 = []
    LS1 = []
    for (x, y) in LS:
        if not y:
            LS0 += [(x, y)]
        else:
            LS1 += [(x, y)]
    LS0 = LS0[0:round(len(LS1)*ratio)]
    LS = LS0 + LS1

    random.shuffle(WS)
    random.shuffle(LS)
    random.shuffle(VS)

    X_WS, y_WS = zip(*WS)
    X_LS, y_LS = zip(*LS)
    X_VS, y_VS = zip(*VS)

    # Model creation and assessment
    models=[]
    with measure_time("Creating and assessing the models"):
        for param1 in [8000]:
            for param2 in ['balanced']:
                for param3 in [None]:
                    for param4 in ["not defined"]:
                        model = RandomForestClassifier(n_estimators=param1,
class_weight=param2, max_depth=param3)
                        model.fit(X_LS, y_LS)

                        y_predict = model.predict_proba(X_VS)[:,1]
                        auc = roc_auc_score(y_VS, y_predict)

                        models += [{"model": model,
                                    "param1": param1,
                                    "param2": param2,
                                    "param3": param3,
                                    "param4": param4,
                                    "auc": auc
                                    }]

    # Finding the best model
    best_model = models[0]
    for model in models:
        if(model["auc"] >= best_model["auc"]):
            best_model = model


    print("------BEST MODEL-------")
    print("param1="+str(best_model["param1"]))
    print("param2="+str(best_model["param2"]))
    print("param3="+str(best_model["param3"]))
    print("param4="+str(best_model["param4"]))
    print("auc="+str(best_model["auc"]))
    print("-----------------------")


    # Best model fitting
    with measure_time("Fitting the best model"):
        best_model["model"].fit(X_WS, y_WS)


    # Changing 0 to 1 ratio in final sample
    WS = list(zip(X_WS, y_WS))
    WS0 = []
    WS1 = []
    for (x, y) in WS:
        if not y:
            WS0 += [(x, y)]
        else:
            WS1 += [(x, y)]
    WS0 = WS0[0:round(len(WS1)*ratio)]
    WS = WS0 + WS1
    random.shuffle(WS)

    # Predicting and submitting
    y_pred = best_model["model"].predict_proba(X_TS)[:,1]

    fname = make_submission(y_pred, best_model["auc"], 'random_forest')
    print('Submission file "{}" successfully written'.format(fname))
    print("Done")

```

Listing 2: Main function

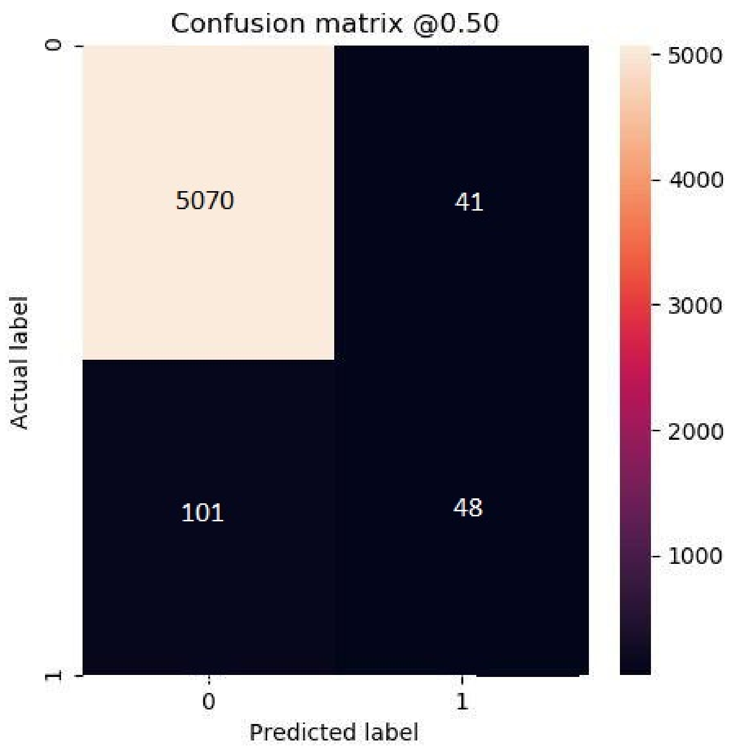
2 Optimization and model validation

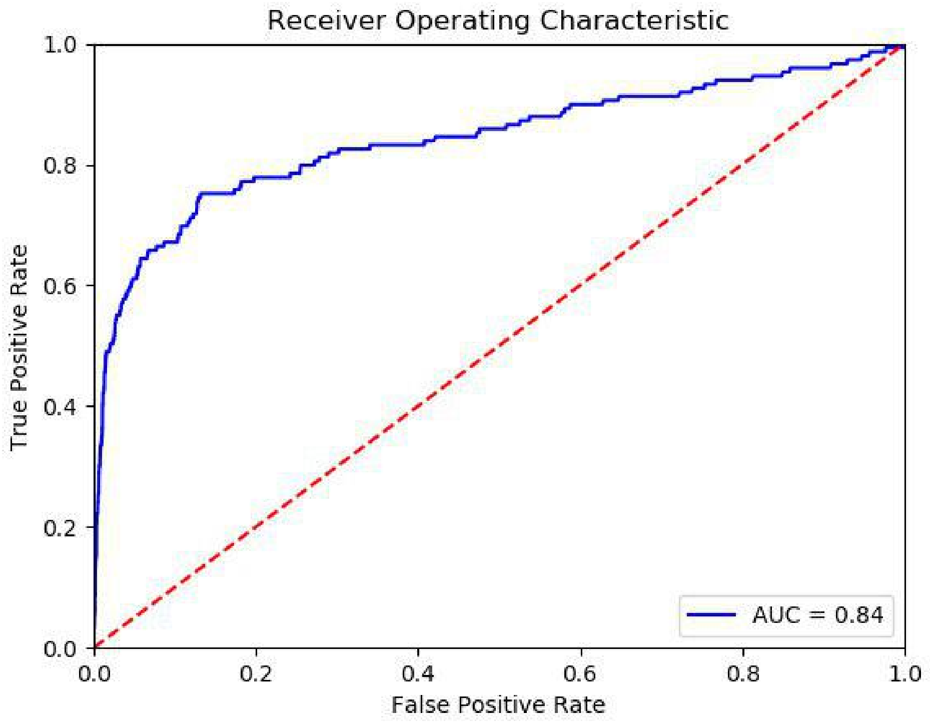
The hyper parameters optimization has been performed using 3 nested `for` loops, one for each parameter to optimize in the `RandomForestClassifier`: `n_estimators`, `class_weight` and `max_depth`. The model has been fitted for each combination the parameters we wanted to try, and for each tested model, we computed the auc score on the validation set. In the end, we kept the highest auc score model as our best model.

3 AUC estimation

The AUC estimation has been obtained using the `roc_auc_score` function on the probability predictions on the validation test. The AUC we obtained was 0.84, while the actual value was 0.77370 on the online 33% set and 0.80270 on the whole online set. We didn't try to predict the auc on the test set to have a more faithful estimation, as we had no time to do so at the end of the challenge and preferred to improve our model. However, the 6% difference with the 33% online set is more or less the difference that we recorded when we submitted other models too.

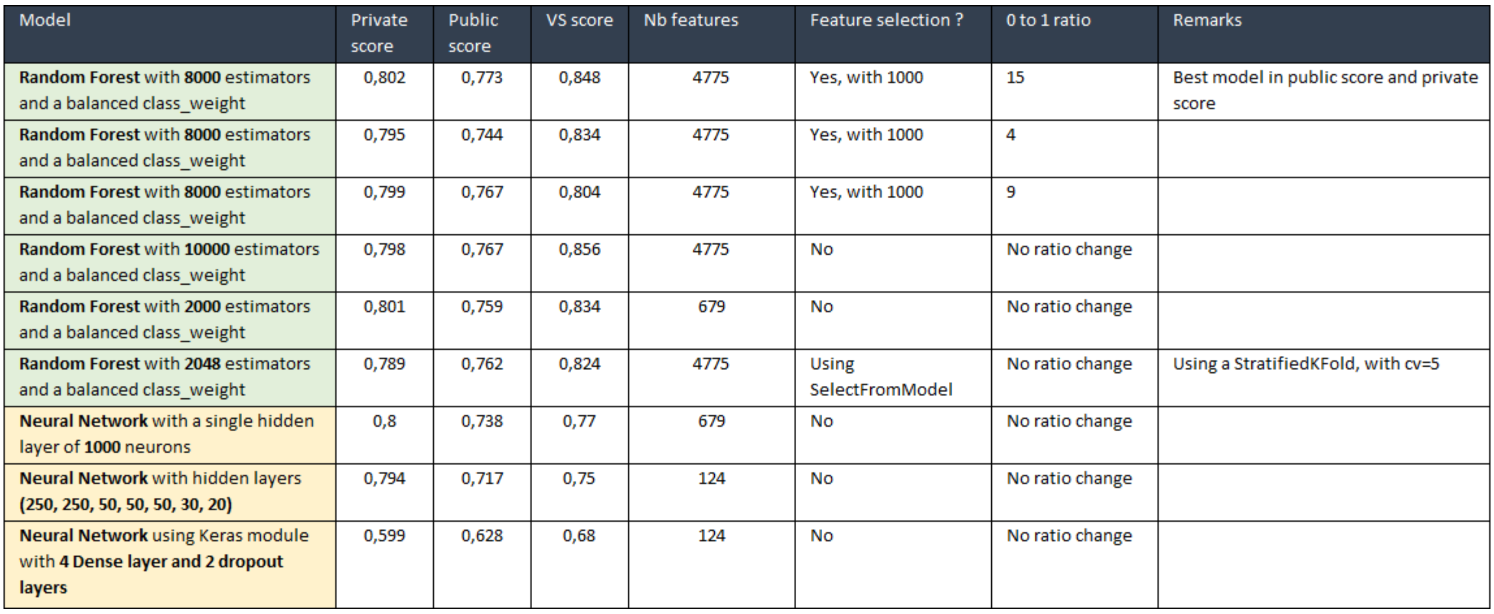
You will find below the confusion matrix and the roc curve of our best model:





4 Other approaches

Here is a summary of some approaches that performed well. Of course not all of the approches we tried are given in the table.



Model	Private score	Public score	VS score	Nb features	Feature selection ?	0 to 1 ratio	Remarks
Random Forest with **8000** estimators and a balanced class_weight	0,802	0,773	0,848	4775	Yes, with 1000	15	Best model in public score and private score
Random Forest with **8000** estimators and a balanced class_weight	0,795	0,744	0,834	4775	Yes, with 1000	4	
Random Forest with **8000** estimators and a balanced class_weight	0,799	0,767	0,804	4775	Yes, with 1000	9	
Random Forest with **10000** estimators and a balanced class_weight	0,798	0,767	0,856	4775	No	No ratio change	
Random Forest with **2000** estimators and a balanced class_weight	0,801	0,759	0,834	679	No	No ratio change	
Random Forest with **2048** estimators and a balanced class_weight	0,789	0,762	0,824	4775	Using SelectFromModel	No ratio change	Using a StratifiedKFold, with cv=5
Neural Network with a single hidden layer of **1000** neurons	0,8	0,738	0,77	679	No	No ratio change	
Neural Network with hidden layers **(250, 250, 50, 50, 50, 30, 20)**	0,794	0,717	0,75	124	No	No ratio change	
Neural Network using Keras module with **4 Dense layer and 2 dropout layers**	0,599	0,628	0,68	124	No	No ratio change	