Activity prediction for chemical compounds

ID2214 Group 6

December 14, 2021

1 Introduction

In this project, our task was to develop a predictive model for chemical compounds, using a suitable representation and learning algorithm, in order to maximize the AUC on the test set. These chemical compounds are represented by text strings using Simplified Molecular Input Line Entry Specification (SMILES), which is a real language structure that uses vocabulary (atomic and bond symbols) and grammatical rules to describe the structure of chemical compounds.

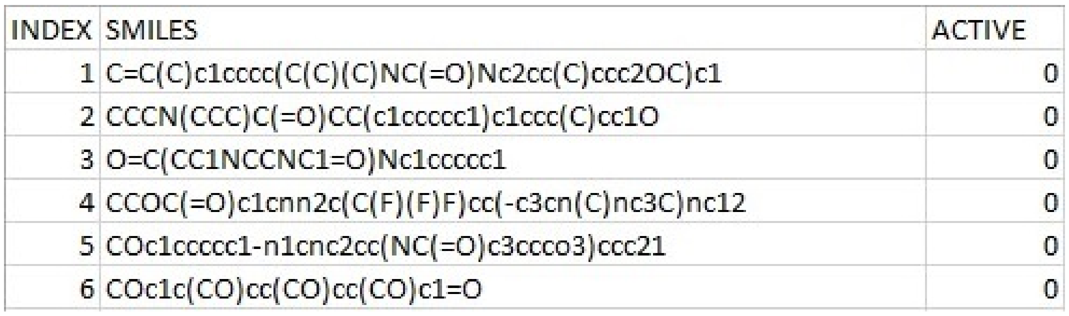
2 Workflow

While working on this project, we have divided our work in three main tasks, which are - choice of feature sets, data preparation, model building and model and parameter selection.

2.1 Choice of feature sets

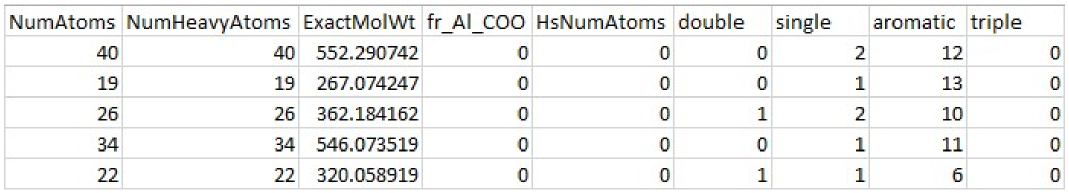
In this part, we have selected features for the training set. We have used all the recommended features using the open source toolkit for cheminformatics, RDKit. We have also used Morgan Fingerprint Vector, which represent presence or absence of substructures. Alongside these features, we have also used Mol2Vec, which is an unsupervised pretraining method to generate an information-rich representation of molecular substructures. Following table contains a glimpse of the features we have used to develop our predictive model.

Feature	Description
Molecular Descriptor	Chem.MolFromSmiles(): Object representing the corresponding chemical compound, for which various properties may be derived. GetNumAtoms(): Number of atoms. rdkit.Chem.Lipinski.HeavyAtomCount(): Number of heavy atoms (Without Hydrogen). rdkit.Chem.rdMolDescriptors.CalcExactMolWt(): Exact molecular weight. GetBondType(): Returns the type of the bond. rdkit.Chem.Fragments.fr$_Al_COO()$: $Numberofaliphaticcarboxylicacids.$
Fingerprint Vector	AllChem.GetMorganFingerprintAsBitVect(x, 2, nBits=124)
Mol2Vec	An unsupervised machine learning approach to learn vector representations of molecular substructures.



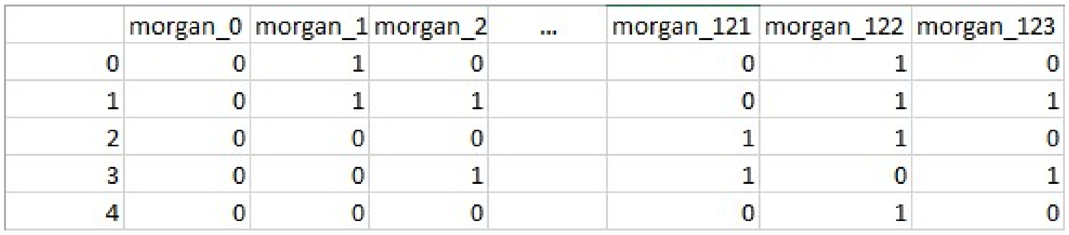
INDEX	SMILES	ACTIVE
1	C=C(C)c1cccc(C(C)(C)NC(=O)Nc2cc(C)ccc2OC)c1	0
2	CCCN(CCC)C(=O)CC(c1ccccc1)c1ccc(C)cc1O	0
3	O=C(CC1NCCNC1=O)Nc1ccccc1	0
4	CCOC(=O)c1cnn2c(C(F)(F)F)cc(-c3cn(C)nc3C)nc12	0
5	COc1ccccc1-n1cnc2cc(NC(=O)c3ccco3)ccc21	0
6	COc1c(CO)cc(CO)cc(CO)c1=O	0

Figure 1: Training data.



NumAtoms	NumHeavyAtoms	ExactMolWt	fr_Al_COO	HsNumAtoms	double	single	aromatic	triple
40	40	552.290742	0	0	0	2	12	0
19	19	267.074247	0	0	0	1	13	0
26	26	362.184162	0	0	1	2	10	0
34	34	546.073519	0	0	0	1	11	0
22	22	320.058919	0	0	1	1	6	0

Figure 2: Training data with features.



	morgan_0	morgan_1	morgan_2	...	morgan_121	morgan_122	morgan_123
0	0	1	0		0	1	0
1	0	1	1		0	1	1
2	0	0	0		1	1	0
3	0	0	1		1	0	1
4	0	0	0		0	1	0

Figure 3: Morgan Fingerprint.

2.2 Data preparation

2.2.1 Tune Parameters

We have used GridSearchCV on training set to tune parameters. The GridSearchCV instance implements the usual estimator API: when "fitting" it on a dataset all the possible combinations of parameter values are evaluated and the best combination is retained.

X = np.array(trainx)

y = np.array(trainy.astype(int))

2.2.2 Split Dataset

The original training dataset is divided into training and test data by 80% and 20%. The datasets are generated using Scikit-learn.

```
from sklearn.model_selection import train_test_split

def split_dataset(traindict, train_y, feature=["Morgan"], ratio=0.2):
    data = pd.DataFrame()
    for f in features:
        data = pd.concat([data, traindict[f]], axis=1)

    X_train, X_test, y_train, y_test = train_test_split(data, train_y, test_size=ratio,
    random_state=10, stratify=train_y)
    return X_train, X_test, y_train, y_test
```

2.3 Model Building

This task aims to building models to predict the activity for chemical compounds. Considering the methods in Scikit-learn, we tried three models as below, and we used different features to find out the performance of models in varying situations.

Model	Reason for choosing
Logistic Regression	Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. Logistic regression is easier to implement, interpret, and very efficient to train.
Random Forest	Random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. It allows quick identification of significant information from vast datasets. The biggest advantage of Random forest is that it relies on collecting various decision trees to arrive at any solution.
Adaboost	AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem.
LightBGM	LightGBM is a fast, distributed, high performance gradient boosting framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks. It has faster training speed and higher efficiency.

2.4 Result and Analysis

Based on the given features, we have run experiments in following situations:

1. Build models based on the features as follows: NoAtoms, CalcExactMolWt, fr_Al_COO and and HeavyAtomCount.
2. Build models based on the features and the morgan fingerprints vector.
3. Build models based on Mol2Vec vector.

After that, we have calculated the AUC using different models. Considering about the AUC score on 5 folds cross-validation and overfitting, we have got following AUC scores:

Features	Logistic Regression	Random Forest	Adaboost	LightGBM
Molecular Features only. No fingerprint vector	0.8155895833307569	0.7719512253921111	0.7872629504749252	0.838577323062901
Molecular Features with morgan fingerprint vector	0.8674449550543966	0.8377176045153915	0.8174422895352957	0.9086116986855508
Molecular Features with morgan fingerprint vector and Mol2Vec	0.8962115675331471	0.8272883331799562	0.8732096433487412	0.9304794459632252

Table 3: AUC Score

Based on the results from Table 3, considering about the AUC score on 5 folds CV and overfitting, we choose lightGBM as our model and molecular features with morgan fingerprint vector and Mol2Vec as our best features. Because even if we combine Random Forest model with Adaboost, the AUC score got beaten by the LightGBM with a fair margin.

2.5 Model and Parameter Selection

2.5.1 Cross-validation

The cross validation is used to see the performance of LightGBM. Here splits is 5, which means we use a 5-fold cross validation. And the number of estimators is 100.

```
fold = KFold(n_splits=5, random_state=40, shuffle=True)
lgbC = LGBMClassifier(n_estimators=100)
cross_val_score(lgbC, trainx, trainy, cv=fold, scoring='roc_auc')

[0.82463594 0.92201499 0.882126 0.88931774 0.87735528]
```

From the results we get an average AUC score of 0.87909059, which is acceptable without optimization.

2.5.2 Parameter selection

We choosed the best parameters for LightGBM using GridSearchCV.

```
parameters = {
            'num_leaves': [60,100],
            'min_data_in_leaf': [40],
            'max_depth': [-5, -1, 5, 10 15, 20, 25, 30, 35],
            "min_sum_hessian_in_leaf": [6],
            'learning_rate': [0.01, 0.02, 0.05, 0.1, 0.15],
            'feature_fraction': [0.6, 0.7, 0.8, 0.9, 0.95],
            'bagging_fraction': [0.8],
            'bagging_freq': [2],
            'lambda_l1': [0.1],
            'cat_smooth': [10]
}
gbm = lgb.LGBMClassifier(boosting_type='gbdt',
                         objective='binary',
                         metric='auc',
                         verbose=-1,
                         num_boost_round=500,
                         random_state=2019
                         )

grid = GridSearchCV(gbm, param_grid=parameters, scoring="roc_auc", verbose=1, cv=5)
grid.fit(X, y)

grid.best_params_

{'bagging_fraction': 0.8, 'bagging_freq': 2, 'cat_smooth': 10, 'feature_fraction': 0.9,
'lambda_l1': 0.1, 'learning_rate': 0.01, 'max_depth': -1, 'min_data_in_leaf': 40,
'min_sum_hessian_in_leaf': 6, 'num_leaves': 60}

grid.best_score_

[0.9304794459632252]
```

3 References

[1] www.rdkit.com
[2] mol2vec.readthedocs.io/en/latest/
[3] www.rdkit.com
[4] https://lightgbm.readthedocs.io/en/latest/
[5] https://scikit-learn.org/stable/modules/ensemble.html