TKO_7092 Evaluation of Machine Learning Methods 2023

Exercise 4

Complete the tasks given to you in the letter below. There are cells at the end of this notebook to which you are expected to write your code. Insert markdown cells as needed to describe your solution. Remember to follow all the general exercise guidelines stated in Moodle.

The deadline of this exercise is **Wednesday 22 February 2023 at 23:59**. Please contact Juho Heimonen (juaheim@utu.fi) if you have any questions about this exercise.

Student name: Jeremias Shadbolt

Student number: 1900386

Student email: jrshad@utu.fi

Dear Data Scientist,

I have a task for you that concerns drug molecules and their targets. I have spent a lot of time in a laboratory to measure how strongly potential drug molecules bind to putative target molecules. I do not have enough resources to measure all possible drug-target pairs, so I would like to first predict their affinities and then measure only the most promising ones. I have already managed to create a model which I believe is good for this purpose. Its details are below.

- algorithm: K-nearest neighbours regressor
- parameters: K=20
- training data: all the pairs for which I have measured the affinity

The data I used to create the model is available in the files `input.data`, `output.data` and `pairs.data` for you to use. The first file contains the features of the pairs, whereas the second contains their affinities. The third file contains the identifiers of the drug and target molecules of which the pairs are composed. The files are paired, i.e. the i*th* row in each file is about the same pair.

I am not able to evaluate how well the model will perform if I use it to predict the affinities of new drug-target pairs. I need you to evaluate the model for me. There are three distinct situations in which I want to use this model in the future.

1. Because I have only measured the affinities for some of the possible pairs of the currently known drugs and targets, I want to use the model to predict the affinities for the remaining pairs.
2. I am confident that I will discover new potential drug molecules in the future, so I will want to use the model to predict their affinities to the currently known targets.
3. Because new putative target molecules, too, will likely be identified in the future, I will also want to use the model to predict the affinities between the drug molecules I will discover and the target molecules somebody else will discover in the future.

Please evaluate the generalisation performance of the model in these three situations. I need to get evaluation results from leave-one-out cross-validation with C-index. Also, because I'm worried that unreliable evaluation results will mislead me to waste precious resources, please explain why I can trust your results.

Yours sincerely, \ Bio Scientist

Import libraries

In [1]:
```
# Import the libraries you need.
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
```

Load datasets

In [2]:
```
# Read the data files (input.data, output.data, pairs.data).
input_data = np.genfromtxt('input.data', dtype=float)
output_data = np.genfromtxt('output.data')
pairs_data = np.genfromtxt('pairs.data', dtype=str)
```

In [3]:
```
# Verify data was read correctly
display(input_data[:1])
display(output_data[:1])
display(pairs_data[:1])
```

```
array([[6.53771, 7.04273, 7.30593, ..., 8.13992, 7.36155, 7.9893 ]])
array([10000.])
array([['"D23"', '"T194"']], dtype='<U6')
```

In [4]:
```
# Standardize the inputs since we're using K-nn
input_data = StandardScaler().fit_transform(input_data)
```

Write functions

In [5]:
```
# Write the functions you need to perform the requested cross-validations.
```

In [6]:
```
# C-index score from previous exercises
def cindex(true_labels, pred_labels):
    n = 0
    h_num = 0
    for i in range(0, len(true_labels)):
        t = true_labels[i]
        p = pred_labels[i]
        for j in range(i+1, len(true_labels)):
            nt = true_labels[j]
            np = pred_labels[j]
            if (t != nt):
                n = n + 1
                if (p < np and t < nt) or (p > np and t > nt):
                    h_num += 1
                elif (p == np):
                    h_num += 0.5
    cindx = h_num  /n
    return cindx
```

1. Because I have only measured the affinities for some of the possible pairs of the currently known drugs and targets, I want to use the model to predict the affinities for the remaining pairs.

this sounds like LOOCV, so let's implement that!

In [7]:
```
def loocv_cindex(X, y, model):
    '''
    LEAVE ONE OUT CROSS VALIDATION
    INPUT:
    X = input data / features
    y = output / labels
    model = ml model to estimate

    OUTPUT:
    C-index score for model
    '''
    # List for predicted values
    predictions = list()

    for i in range(len(X)):
        # Choose the ith row as test set and leave ith row out from train sets
        X_test = X[i].reshape(1, -1)
        X_train = np.delete(X, i, axis = 0)
        y_train = np.delete(y, i, axis = 0)
        model.fit(X_train, y_train)
        predictions.append(model.predict(X_test))

    return cindex(y, predictions)
```

1. I am confident that I will discover new potential drug molecules in the future, so I will want to use the model to predict their affinities to the currently known targets.

Leave-Target-Out CV

In [8]:
```
def ltocv_index_target(X, y, targets, model):
    '''
    LEAVE TARGET OUT CROSS VALIDATION
    INPUT:
    X = input data / features
    y = output data / labels
    targets = targets' column in data
    model = model
    '''
    predictions = list()
    for i in range(len(X)):
        # Save the ith target for train set selection
        target = targets[i]
        # Choose the ith row as test set
        X_test = X[i].reshape(1, -1)
        # Choose train sets from rows that doesn't contain the target
        X_train = X[(target != targets)]
        y_train = y[(target != targets)]

        model.fit(X_train, y_train)
        predictions.append(model.predict(X_test))

    return cindex(y, predictions)
```

1. Because new putative target molecules, too, will likely be identified in the future, I will also want to use

the model to predict the affinities between the drug molecules I will discover and the target molecules somebody else will discover in the future.

LPOCV but slightly modified

In [9]:
```
def lpocv_cindex_identify(X, y, pairs, model):
    '''
    LEAVE PAIR OUT CROSS VALIDATION which calculates C-index when known targets and identi
    INPUT:
    X = input
    y = labels
    pairs = identifiers and targets as pairs
    model = model
    '''
    predictions = list()
    # Extract columns from pairs
    a_cols = pairs[:, 0]
    b_cols = pairs[:, 1]
    for i in range(len(X)):
        # Save the ith pair of elements for train set selection
        pair = pairs[i]
        # Choose the ith row as test set
        X_test = X[i].reshape(1, -1)
        # Choose train sets from rows that doesn't have either of the elements in our ith
        X_train = X[(pair[0] != a_cols) & (pair[0] != b_cols) & (pair[1] != a_cols) & (pai
        y_train = y[(pair[0] != a_cols) & (pair[0] != b_cols) & (pair[1] != a_cols) & (pai

        model.fit(X_train, y_train)
        predictions.append(model.predict(X_test))

    return cindex(y, predictions)
```

Run cross-validations

In [10]:
```
# Run the requested cross-validations and print the results.

model = KNeighborsRegressor(n_neighbors = 20, metric='euclidean')

loocv_result = loocv_cindex(input_data, output_data, model)
print(f"C-index score for predicting affinities of remaining pairs: {loocv_result:.2f}")

ltocv_target_result = ltocv_index_target(input_data, output_data, pairs_data[:, 1], model)
print(f"C-index score for predicting affinities to known targets: {ltocv_target_result:.2f

lpocv_identify_result = lpocv_cindex_identify(input_data, output_data, pairs_data, model)
print(f"C-index score for predicting affinities to unknown targets: {lpocv_identify_result
```

```
C-index score for predicting affinities of remaining pairs: 0.78
C-index score for predicting affinities to known targets: 0.77
C-index score for predicting affinities to unknown targets: 0.68
```

Interpret results

In [11]:
```
# Interpret the results you obtained and explain why they can be trusted.
```

The results seem good enough. Accuracies of 78 % and 77 % would imply that the model generalizes well to unseen data and it's not overfit. I believe with some hyperparameter tuning (n_neighbours and others) we could probably reach even better results, but for a 'vanilla K-nn' this seems ok. The model which predicted affinities to unknown targets could probably use some tuning since 68 % isn't very good.

Leave-one-out was the CV method to go for here since we wanted to use all available data and the amount of data allowed us to do so.