Ersilia

H3D FOUNDATION

H3D Foundation and Ersilia present

Bringing data science and AI/ML tools to infectious disease research

Event Sponsors

wellcome

CS&S

Code for Science & Society

Session 2: Building a machine learning model

Skills Development workshop

Gemma Turon, @TuronGemma, gemma@ersilia.io
Miquel Duran-Frigola, @mduranfrigola, miquel@ersilia.io
Ersilia Open Source Initiative, @ersiliaio, https://ersilia.io
28th September 2022

Ersilia

H3D FOUNDATION

Tools

Google Colaboratory

Therapeutics Data Commons

Python Packages

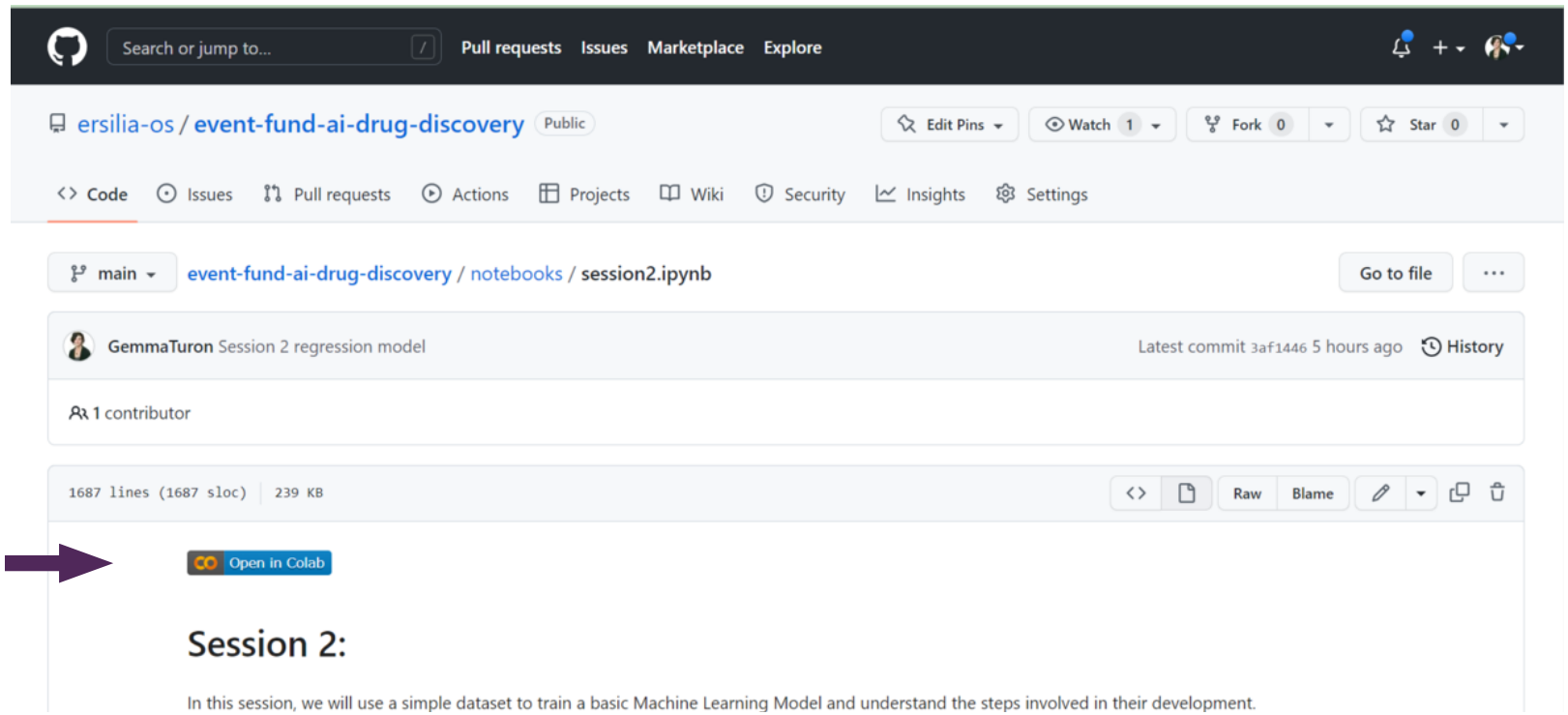
Chemical Checker

scikit learn mach

RDKit

Getting Started

https://github.com/ersilia-os/event-fund-ai-drug-discovery



Classification tasks

A classifier identifies the category of a new data point. In the case of bioactivity data, we usually find two categories (binary classifier)

- Active (1)
- Inactive (0)

To learn more about classification, we will use the Therapeutics Data Commons prepared dataset for blockade of hERG activity (Tox)

- Actives (1): block hERG
- Inactives (0): do not block hERG

Wang, Shuangquan et al, Molecular Pharmaceutics 2016

Load the data from TDC

▸ Download TDC hERG Dataset

Click on the play button to get the train and tests sets for the TDC hERG data By running this cell, you will get the datasets saved to the drive in data/session2

Show code

```
# import the hERG dataset, part of the Toxicity data available at TDC
!pip install PyTDC

from tdc.single_pred import Tox
data = Tox(name = "hERG")

split = data.get_split()

#we can now separate the compressed dataset in the three sections
train = split["train"]
validation = split["valid"]
test = split["test"]

train.to_csv("drive/MyDrive/h3d_ersilia_ai_workshop/data/session2/tdc_herg_train.csv", index=False)
validation.to_csv("drive/MyDrive/h3d_ersilia_ai_workshop/data/session2/tdc_herg_validation.csv", index=False)
test.to_csv("drive/MyDrive/h3d_ersilia_ai_workshop/data/session2/tdc_herg_test.csv", index=False)
```

Ersilia

H3D FOUNDATION

Load the data from TDC

```
#we can check how many molecules we have in each set
print(len(train))
print(len(valid))
print(len(test))
```

```
458, 66, 131
```

the train set is used to train the ML model	the validation set is used to improve the model	the test set is used to evaluate the final model performance

In this example we will not focus on optimizing the model parameters, therefore we will only use the train and test splits

Ersilia

H3D FOUNDATION

Load the data from TDC

```
#we can check how many molecules we have in each set
print(len(train))
print(len(valid))
print(len(test))
```

```
458, 66, 131
```

the train set is used to train the ML model	the validation set is used to improve the model	the test set is used to evaluate the final model performance

```
train.head()
print(len(train[train["Y"]==0]))
print(len(train[train["Y"]==1]))
```

	Drug_ID	Drug	Y
0	DEMETHYLASTEMIZOLE	Oc1ccc(CCN2CCC(Nc3nc4ccccc4n3Cc3ccc(F)cc3)CC2)cc1	1.0
1	GBR-12909	Fc1ccc(C(OCC[NH+]2CC[NH+](CCCc3ccccc3)CC2)c2cc...	1.0
2	CLOFILIUM PHOSPHATE	CCCCCCC[N+](CC)(CC)CCCCc1ccc(Cl)cc1.CCCCCCC[N+...	1.0
3	FLUSPIRILENE	O=C1NCN(c2ccccc2)C12CC[NH+](CCCC(c1ccc(F)cc1)c...	1.0
4	VANOXERINE HYDROCHLORIDE	Fc1ccc(C(OCCN2CCN(CCCc3ccccc3)CC2)c2ccc(F)cc2)cc1	1.0

Ersilia

Molecular Representation

```
!pip install rdkit  #chemioinformatics package
from rdkit import Chem
from rdkit.Chem import Draw

smiles = train["Drug"][449:] # select a few smiles for representation
#convert the smiles to RdKit Molecules (not readable by humans)
mols = [Chem.MolFromSmiles(smi) for smi in smiles]
```

```
<rdkit.Chem.rdchem.Mol at 0x7fa16c9435d0>
```

```
from rdkit.Chem import Draw
Draw.MolsToGridImage(mols)
```

Ersilia

H3D FOUNDATION

Molecular Featurization

Featurization is the conversion of SMILES into vectors or images that can be passed to the ML model

```
!pip install signaturizer
from signaturizer import Signaturizer
sign = Signaturizer("GLOBAL")
X_train = sign.predict(train["Drug"]).signature
X_test = sign.predict(test["Drug"]).signature
```

```
X_train[0]
```

```
array([-0.09308645,  0.09305048, -0.09294817, ..., -0.05613168,
       -0.1379203 ,  0.05799835], dtype=float32)
```

Duran-Frigola et al, Nat Biotech 2020

Data we have so far:

Original datasets: table with SMILES and its associated hERG blocking activity as 0 or 1

- train, test

X datasets: vector-like representation of the molecules calculated using the Chemical Checker --> model input

- X_train, X_test

Y_datasets: list of 0 and 1 results that correspond to the list of molecules in the train and test sets

- Y_train, Y_test

Supervised Machine Learning

In summary, we now have two datasets (train, test):

- X (input): molecular signature
- Y (output): activity against hERG, 1 or 0

We will use the Random Forest Classifier algorithm implemented in scikit-learn.

```
!pip install sklearn
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train, Y_train)
```

Classification outputs

Once the model is fitted, we can predict the category of each molecule in the validation and test sets. A classifier outputs two numbers per each prediction:

- Probability of 0
- Probability of 1

```
1 y_pred = clf.predict_proba(X_test)
2 y_pred[:10]
```

```
1 y_pred[:,1]
```

```
array([[0.31, 0.69],
       [0.37, 0.63],
       [0.69, 0.31],
       [0.14, 0.86],
       [0.58, 0.42],
       [0.84, 0.16],
       [0.163,0.837],
       [0.257, 0.743],
       [0.111, 0.889],
       [0.303, 0.697]])
```

Classification outputs

In a classification, the output is a probability. We need to define a cut-off or threshold to transform the results into a binary output (0 or 1) again. The threshold is typically set at 0.5 by default.

Proba 0	Proba 1	Cut-off 0.5	Cut-off 0.7	Cut-off 0.3
0.39	0.61			
0.3	0.7			
0.69	0.31			
0.21	0.79			

Classification outputs

In a classification, the output is a probability. We need to define a cut-off or threshold to transform the results into a binary output (0 or 1) again. The threshold is typically set at 0.5 by default.

Proba 0	Proba 1	Cut-off 0.5	Cut-off 0.7	Cut-off 0.3
0.39	0.61	1	0	1
0.3	0.7			
0.69	0.31			
0.21	0.79			

Classification outputs

In a classification, the output is a probability. We need to define a cut-off or threshold to transform the results into a binary output (0 or 1) again. The threshold is typically set at 0.5 by default.

Proba 0	Proba 1	Cut-off 0.5	Cut-off 0.7	Cut-off 0.3
0.39	0.61	1	0	1
0.3	0.7	1	1	1
0.69	0.31			
0.21	0.79			

Classification outputs

In a classification, the output is a probability. We need to define a cut-off or threshold to transform the results into a binary output (0 or 1) again. The threshold is typically set at 0.5 by default.

Proba 0	Proba 1	Cut-off 0.5	Cut-off 0.7	Cut-off 0.3
0.39	0.61	1	0	1
0.3	0.7	1	1	1
0.69	0.31	0	0	1
0.21	0.79			

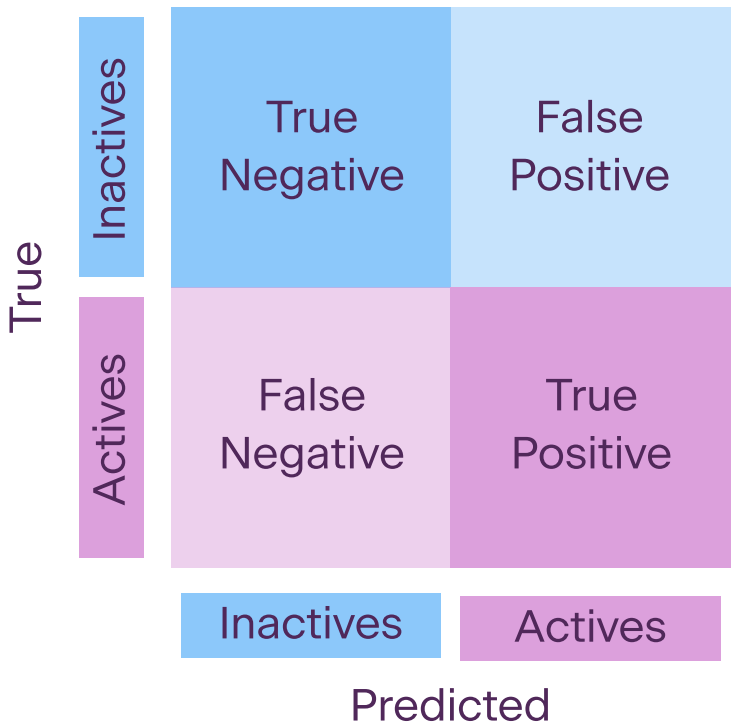
Classification outputs

In a classification, the output is a probability. We need to define a cut-off or threshold to transform the results into a binary output (0 or 1) again. The threshold is typically set at 0.5 by default.

Proba 0	Proba 1	Cut-off 0.5	Cut-off 0.7	Cut-off 0.3
0.39	0.61	1	0	1
0.3	0.7	1	1	1
0.69	0.31	0	0	1
0.21	0.79	1	1	1

Model Evaluation

Confusion Matrix



Precision: how many positives are actually positive

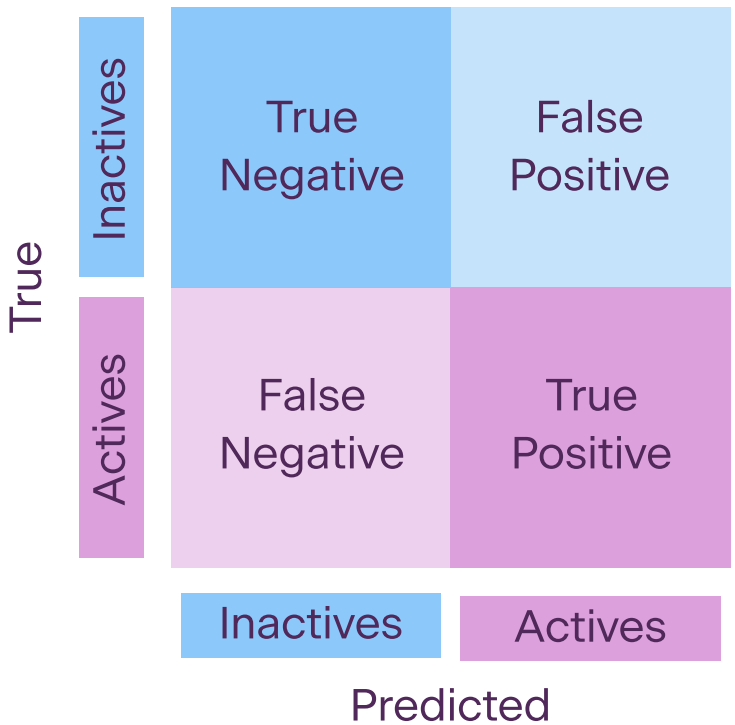
$$\frac{TP}{TP+FP}$$

Recall: how many positives are we able to identify

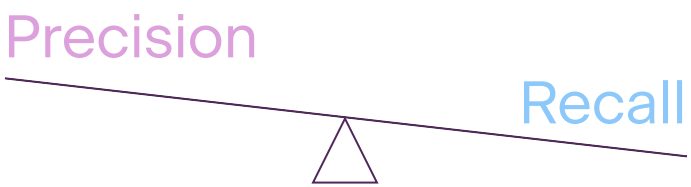
$$\frac{TP}{TP+FN}$$

Model Evaluation

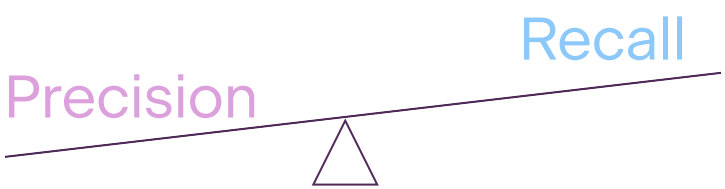
Confusion Matrix



Higher, more restrictive threshold in proba1

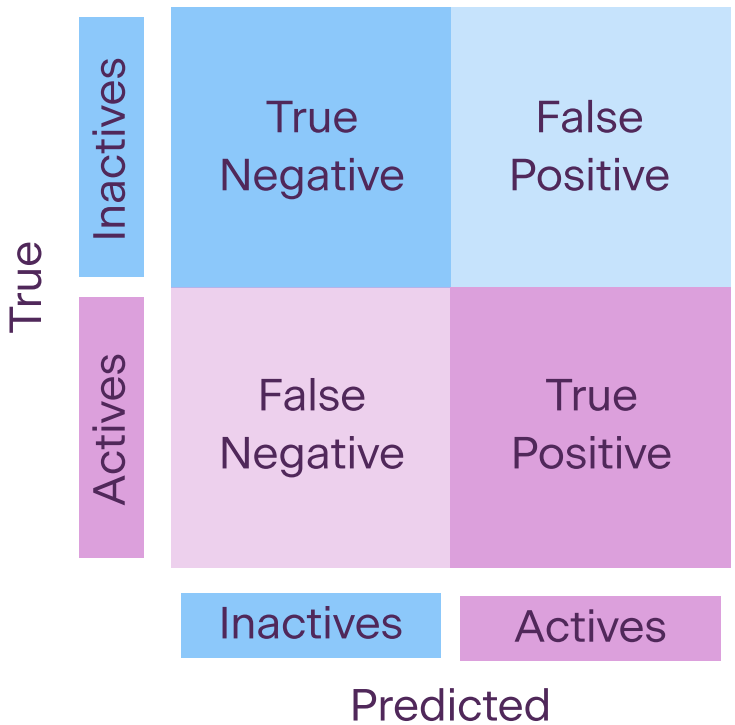


Lower threshold in proba1



Model Evaluation

Confusion Matrix

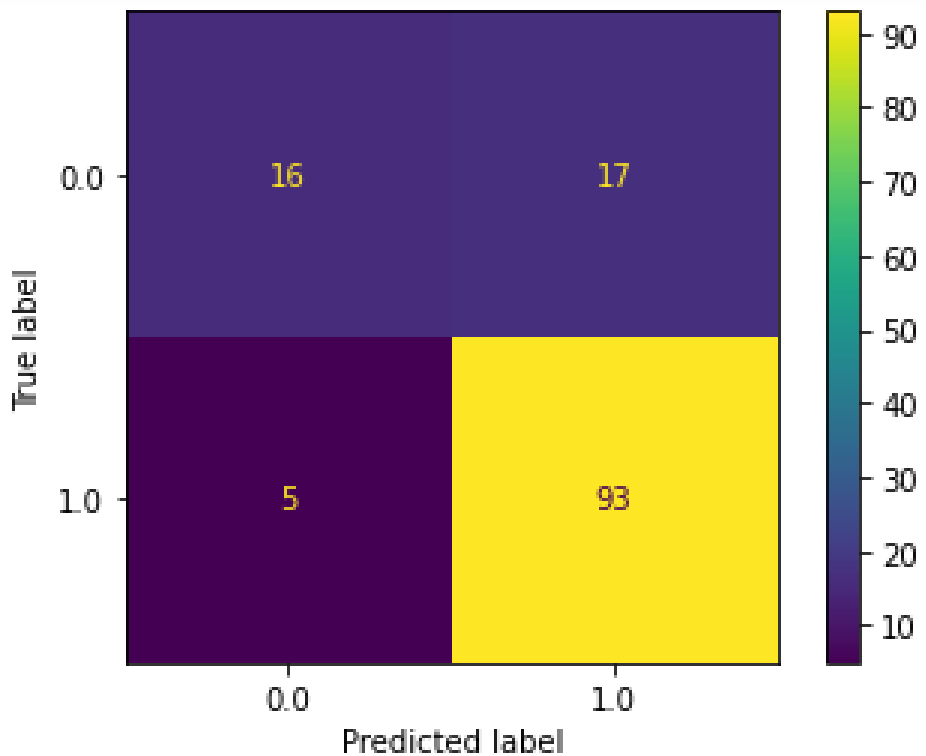


In which scenarios we might want a high precision?

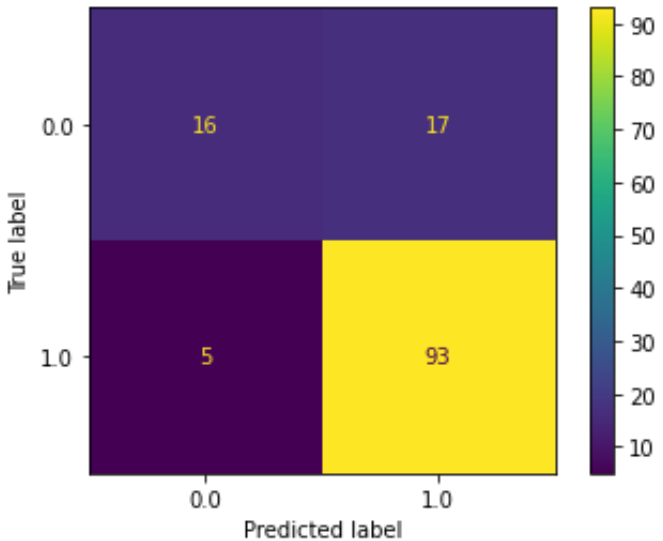
In which scenarios we might want a high recall?

Model Evaluation

```
1 from sklearn.metrics import ConfusionMatrixDisplay as cdm
2 cdm.from_estimator(clf, X_test, Y_test)
```



Model Evaluation



Precision: how many positives are actually positive

Recall: how many positives are we able to identify

$$\frac{TP}{TP+FP} \quad \frac{93}{93+17}$$

$$\frac{TP}{TP+FN} \quad \frac{93)}{93)+5}$$

Ersilia

Model Evaluation

```
from sklearn.metrics import precision_score, recall_score
#precision and recall scores need a specific threshold

y_pred_bin = []
for x in y_pred[:,1]:
    if x > 0.5:
        y_pred_bin += [1]
    if x <= 0.5:
        y_pred_bin += [0]

precision = precision_score(Y_test, y_pred_bin)
recall = recall_score(Y_test, y_pred_bin)
print(precision, recall)
```

```
0.8454545454545455 0.9489795918367347
```

Data we have so far:

Original datasets: train, test

X datasets: X_train, X_test

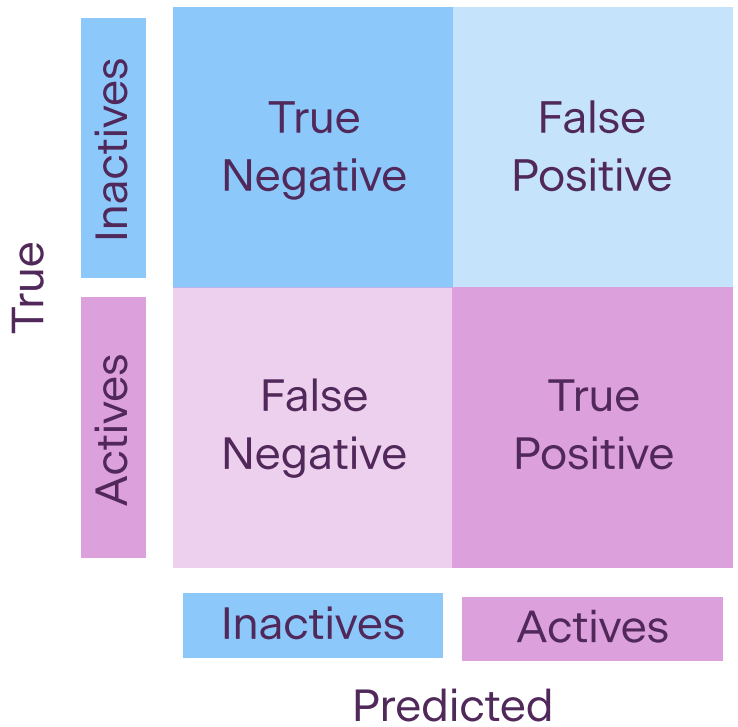
Y_datasets: Y_train, Y_test

Predictions:

- y_pred: output of the classifier, expressing the probability that a molecule is inactive (0) or active (1).
- y_pred_bin: binarized activity based on the predicted probability, the cut-off is set by the researcher.

Model Evaluation

ROC Curve: performance of the model at all classification thresholds (0-1)

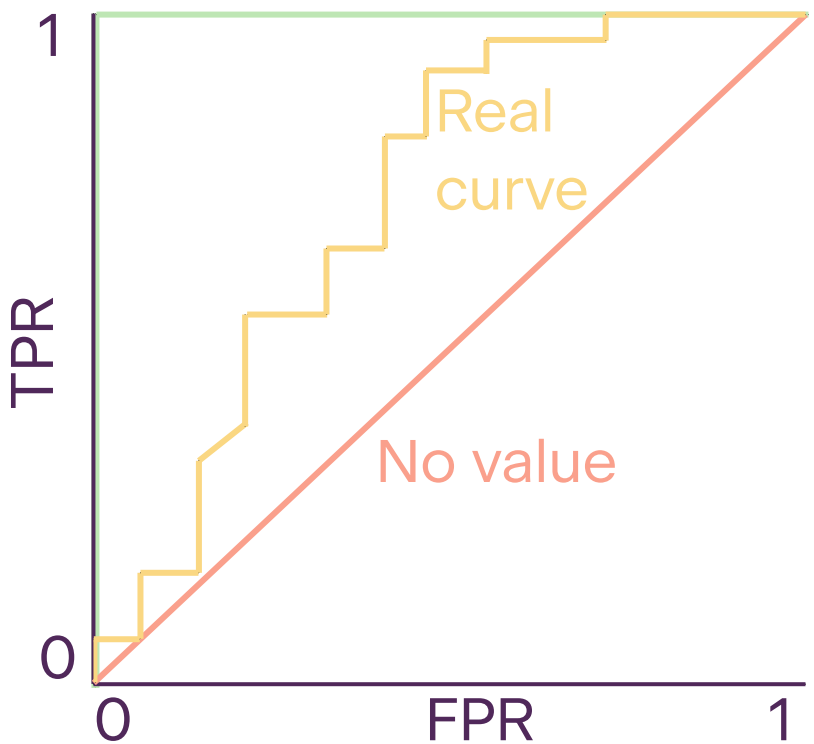


True Positive Rate (sensitivity or recall): proportion of correctly predicted positives of all positive observations (TP/(TP+FN))

False Positive Rate (100-sensibility): proportion of incorrectly predicted positives of all negative observations (FP/(TN+FP))

Model Evaluation

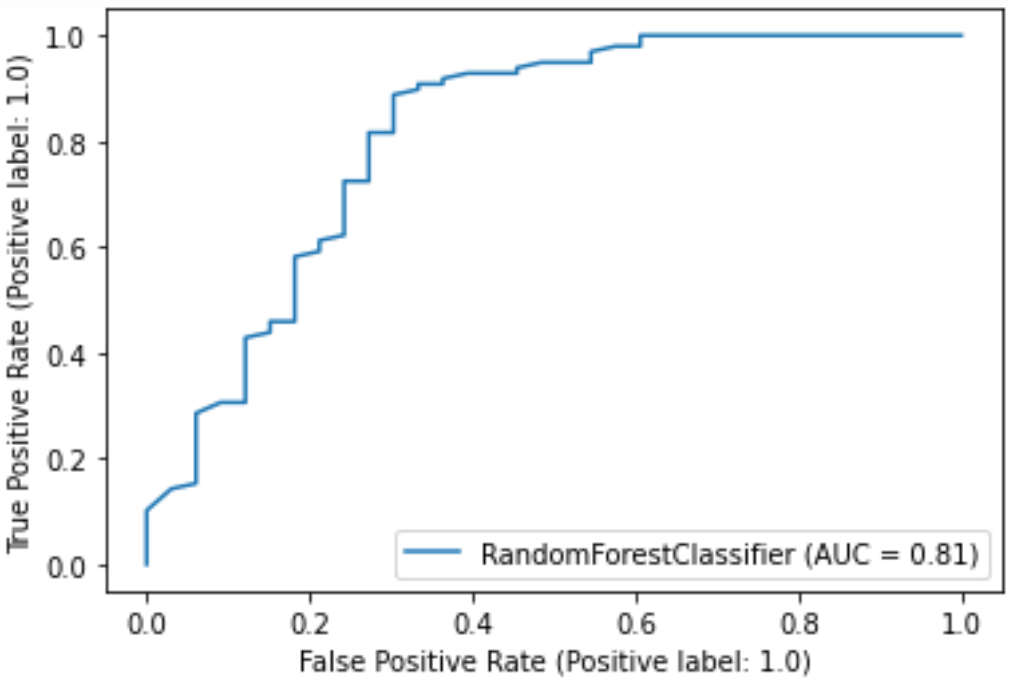
ROC Curve: performance of the model at all classification thresholds (0-1)



Area Under the Curve (AUC): agregate measure of model performance. It does not depend on the threshold

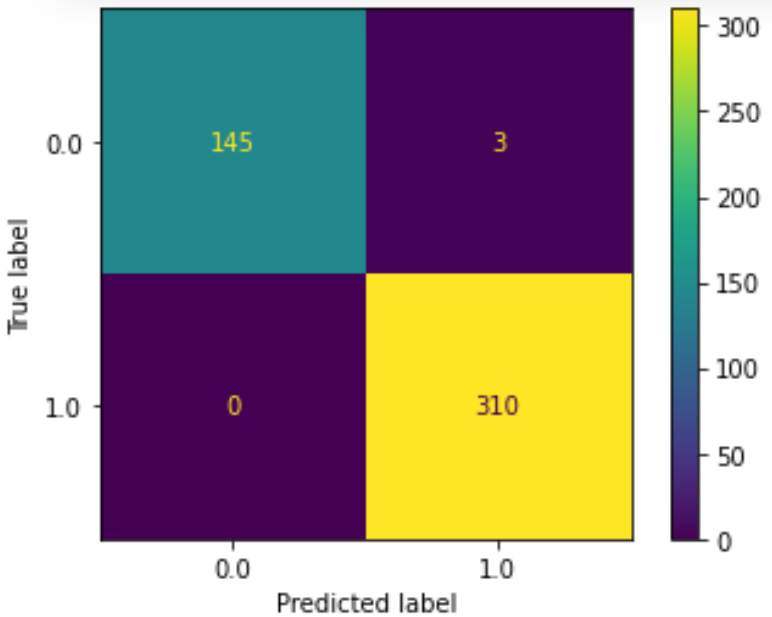
Model Evaluation

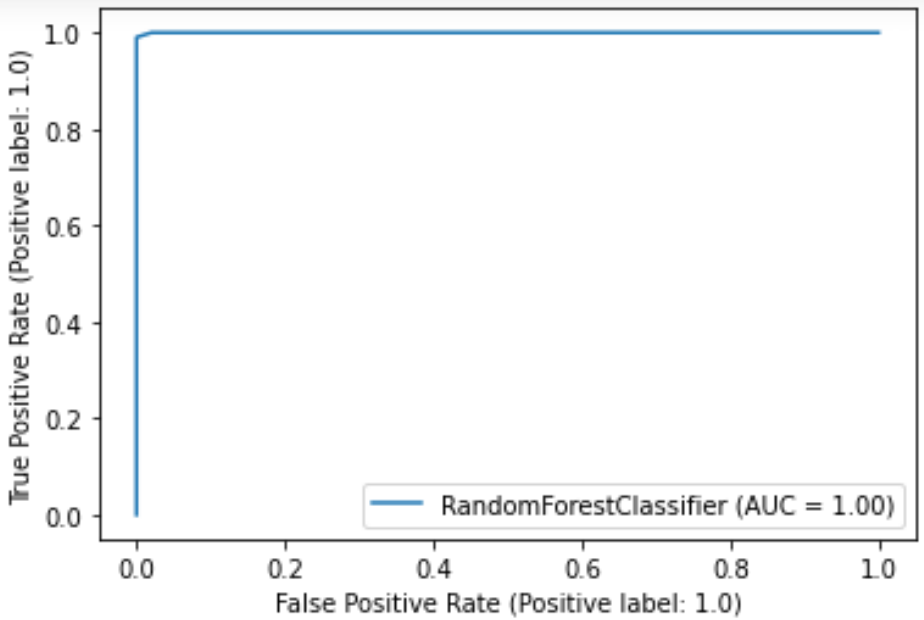
```
1 from sklearn.metrics import RocCurveDisplay as rdc
2 rdc.from_estimator(clf, X_test, Y_test)
```



Model Evaluation

```
1 from sklearn.metrics import ConfusionMatrixDisplay as cdm
2 cdm.from_estimator(clf, X_train, Y_train)
3
4 from sklearn.metrics import RocCurveDisplay as rdc
5 rdc.from_estimator(clf, X_train, Y_train)
```





Classification tasks: summary

- We have used a binary classification for hERG activity and trained a random forest classifier
- The molecules were featurized using Chemical Checker signatures, which combine structural and bioactivity data of the molecules
- The model performance was good (AUC above 0.8) on both the validation and test sets
- How could we try to improve the model?

Regression tasks

A regressor predicts continuous numerical values, for example:

- IC50
- % of inhibition

To learn more about regression, we will use the Therapeutics Data Commons Acute Toxicity:

- Lethal Dose 50 (LD50): amount of drug (mg/kg) that kills 50% of the tested population

Zhu, Hao et al, Chemical Research in toxicology 2009

Load the data from TDC

Download TDC LD50 Dataset

Click on the play button to get the train and tests sets for the TDC hERG data By running this cell, you will get the datasets saved to the drive in data/session2

Show code

```
1
2 #we can check how many molecules we have in each set
3 print(len(train))
4 print(len(valid))
5 print(len(test))
```

Train set: 5170 molecules	Validation set: 738 molecules	Test set: 1477 molecules

Ersilia

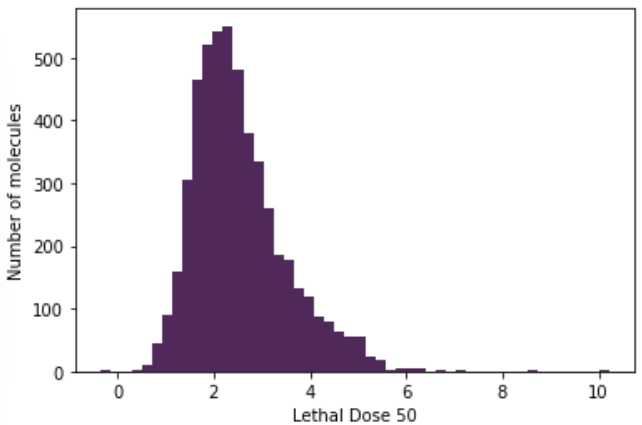
H3D FOUNDATION

Molecular Representation

```
1 train.head()
```

	Drug_ID	Drug	Y
0	Methane, tribromo-	BrC(Br)Br	2.343
1	Bromoethene (9CI)	C=CBr	2.330
2	1,1'-Biphenyl, hexabromo-	Brc1ccc(-c2ccc(Br)c(Br)c2Br)c(Br)c1Br	1.465
3	Isothiocyanic acid, p-bromophenyl ester	S=C=Nc1ccc(Br)cc1	2.729
4	Benzene, bromo-	Brc1ccccc1	1.765

```
1 import matplotlib.pyplot as plt
2
3 plt.hist(train["Y"], bins=50, color
  = "#50285a")
4 plt.xlabel("Lethal Dose 50")
5 plt.ylabel("Number of molecules")
```



Ersilia

H3D FOUNDATION

Molecular Featurization

Chemical Checker Signatures

```python
from signaturizer import Signaturizer
sign = Signaturizer("GLOBAL")
X_train = sign.predict(train["Drug"]).signature
X_test = sign.predict(test["Drug"]).signature
```

```
array([-0.09308645,  0.09305048, -0.09294817, ..., -0.05613168,
       -0.1379203 ,  0.05799835], dtype=float32)
```

```python
Y_train = list(train["Y"])
Y_test = list(test["Y"])
```

```
2.343
```

Duran-Frigola et al, Nat Biotech 2020

Supervised Machine Learning

In summary, we now have three datasets (train, validation, test):

- X (input): molecular signature
- Y (output): LD50 values

We will use a simple linear regression from sci-kit learn to train a model

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, Y_train)
```

Regression outputs

The regression gives back the predicted value, in our case, LD50.

```
y_pred = reg.predict(X_test)

#if we check, the prediction is simply a list of LD50
y_pred[:4]
```

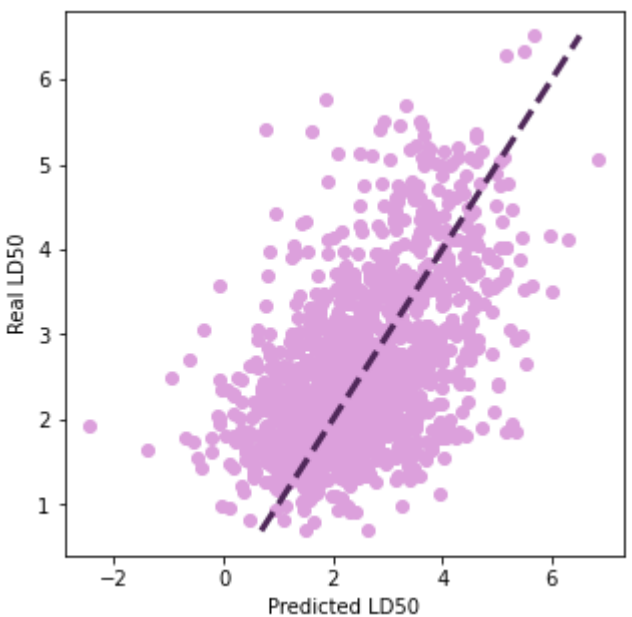
```
array([3.3221316, 2.0938659, 2.874437 , 4.17527  ], dtype=float32)
```

Regression model evaluation

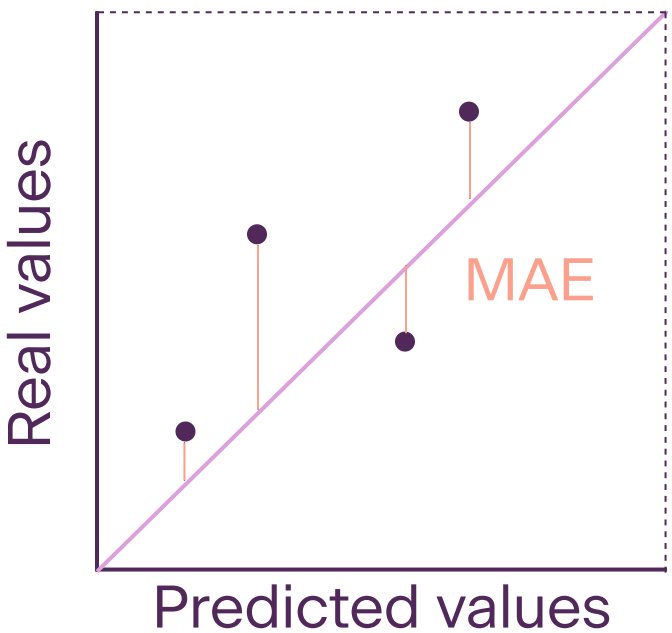
The easiest is to plot a scatter distribution of the real versus the predicted values.

We use the Matplotlib package for plotting functions

```
1 fig, ax = plt.subplots(1,1, figsize=
  (5,5))
2 ax.scatter(x=y_pred, y=Y_test,
  color="#dca0dc")
3 ax.set_xlabel("Predicted LD50")
4 ax.set_ylabel("Real LD50")
```



Regression Model Evaluation



- Mean Absolute Error (MAE): average of the absolute difference between actual and predicted values
- Mean Squared Error (MSE): average of the squared difference
- R-square: proportion of the variance explained by the regression model. It is scale free

Ersilia

Regression model evaluation

```
1 from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
2
3 mae = mean_absolute_error(Y_valid, valid_pred)
4 mse = mean_squared_error(Y_valid, valid_pred)
5 r2 = r2_score(Y_valid, valid_pred)
6
7 print("MAE: {}".format(mae))
8 print("MSE: {}".format(mse))
9 print("R2: {}".format(r2))
```

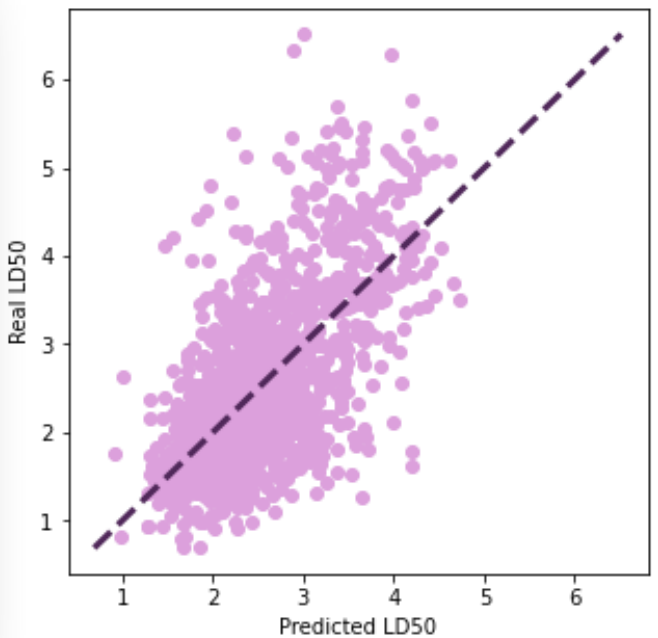
```
MAE: 0.8202228248986185
MSE: 1.1181765214543737

R2: -0.2515852592994649 6
```

What do we think of these results?

Improving regression models

We can test other models for regression to improve the performance on our data

```
1  from sklearn.ensemble import
   RandomForestRegressor
2
3  reg = RandomForestRegressor(n_estimators=10)
   #we use only 10 trees for speed time
4  reg.fit(X_train, Y_train)
5
6  test_pred = reg.predict(X_test)
7
8  fig, ax = plt.subplots(1,1, figsize=(5,5))
9  ax.scatter(x=y_pred, y=Y_test,
   color="#dca0dc")
10 ax.set_xlabel("Predicted LD50")
11 ax.set_ylabel("Real LD50")
```

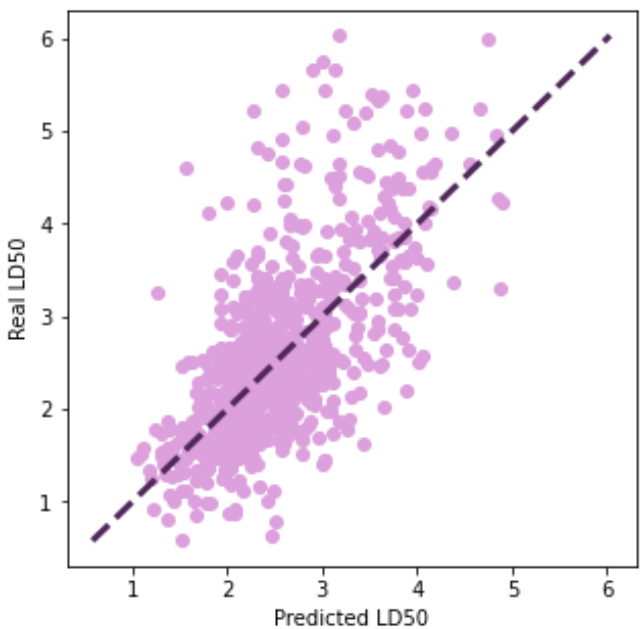


Improving regression models

We can test other models for regression to improve the performance on our data

```
1 mae = mean_absolute_error(Y_valid, valid_pred)
2 mse = mean_squared_error(Y_valid, valid_pred)
3 r2 = r2_score(Y_valid, valid_pred)
4
5 print("MAE: {}".format(mae))
6 print("MSE: {}".format(mse))
7 print("R2: {}".format(r2))
```

```
MAE: 0.5450362715297566
MSE: 0.5291154421053693

R2: 0.40775622175874715
```



Conclusions

- We have used the Therapeutics Data Commons datasets to build simple classification and regression QSAR models:
  - Random Forest Classifier
  - Linear Regression
  - Random Forest Regressor
- We have evaluated model performance using the validation and test sets, and the basic metrics for:
  - Classification: precision, recall and AUC
  - Regression: MAE, MSE, R2