In [64]:
```
import numpy as np
import pandas as pd
from rdkit import Chem
from rdkit.Chem import Descriptors
from rdkit.Chem.MolStandardize import rdMolStandardize
from rdkit.ML.Descriptors import MoleculeDescriptors as md
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
import xgboost as xgb
import matplotlib.pyplot as plt
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    accuracy_score, f1_score, precision_score, recall_score,
    roc_auc_score, confusion_matrix, classification_report, roc_curve
)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [65]:
```
def molecule_from_smiles(smiles):
    try:
        # Extract molecule
        molecule = Chem.MolFromSmiles(smiles, sanitize=True)
        if molecule is None:
            return None, "failed"

        # Remove salts
        clean_molecule = rdMolStandardize.LargestFragmentChooser()
        molecule = clean_molecule.choose(molecule)

        # Sanitize molecule again to reflect changes
        Chem.SanitizeMol(molecule)
        return molecule, "succeed"
    except Exception as e:
        return None, f"error: {e}"


def calculate_descriptors(molecule):
    # Get all descriptors (1D/2D)
    descriptor_names = []
    for descriptor, _ in Descriptors._descList:
        descriptor_names.append(descriptor)

    # Use descriptors to calculate values
    calculator = md.MolecularDescriptorCalculator(descriptor_names)
    descriptor_values = calculator.CalcDescriptors(molecule)

    # Create dictionary
    descriptors = dict(zip(descriptor_names, descriptor_values))
    return descriptors
```

In [66]:
```
dataset = pd.read_excel("in_chemico_dataset.xlsx", engine="openpyxl", ski

descriptor_rows = []
state_molecules = []
molecules = []
```

```
for smiles in dataset["SMILES code"].astype(str):
    # Get molecule
    molecule, state = molecule_from_smiles(smiles)
    state_molecules.append(state)
    molecules.append(molecule)

    # Calculate
    if molecule is None:
        descriptor_rows.append({})
        continue

    descriptors = calculate_descriptors(molecule)
    descriptor_rows.append(descriptors)

descriptor_data = pd.DataFrame(descriptor_rows)

numeric_columns = []
for col in descriptor_data.columns:
    if pd.api.types.is_numeric_dtype(descriptor_data[col]):
        numeric_columns.append(col)

clean_desc = descriptor_data[numeric_columns].copy()

for col in clean_desc.columns:
    clean_desc[col] = clean_desc[col].replace([np.inf, -np.inf], np.nan)

# Drop columns with any NaN
cols_with_nan = []
for col in list(clean_desc.columns):
    if clean_desc[col].isna().any():
        cols_with_nan.append(col)

if len(cols_with_nan) > 0:
    clean_desc = clean_desc.drop(columns=cols_with_nan)

constant_cols = []
for col in list(clean_desc.columns):
    if clean_desc[col].nunique(dropna=False) <= 1:
        constant_cols.append(col)


if len(constant_cols) > 0:
    clean_desc = clean_desc.drop(columns=constant_cols)

output = pd.concat([dataset.reset_index(drop=True), descriptor_data.reset
output["MoleculeStatus"] = state_molecules

# Save
with pd.ExcelWriter("in_chemico_dataset_processed.xlsx", engine="openpyxl
    # Full report with raw 1D/2D descriptors
    output.to_excel(writer, index=False, sheet_name="Raw_Descriptors")

    # Clean X for XGBoost
    clean_desc.to_excel(writer, index=False, sheet_name="X_1D2D_clean")

clean_desc.to_csv("X_1D2D_descriptors.csv", index=False)

print(f"Rows: {len(output)}/Columns in raw: {output.shape[1]}")
print(f"Columns in X (clean): {clean_desc.shape[1]}")
print("First rows of X:")
```

```
print(clean_desc.head().to_string(index=False))
```

```
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: C=C[C@H]1CN2CC[C@H]1C[C@H]2[C@H](O)c1ccnc2ccc(OC)cc12
[02:34:29] New largest fragment: C=C[C@H]1CN2CC[C@H]1C[C@H]2[C@H](O)c1ccnc
2ccc(OC)cc12 (48)
[02:34:29] Fragment: Cl
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: CN(C)CCCN1c2ccccc2Sc2ccc(Cl)cc21
[02:34:29] New largest fragment: CN(C)CCCN1c2ccccc2Sc2ccc(Cl)cc21 (40)
[02:34:29] Fragment: Cl
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: CC1c2cccc(O)c2C(O)=C2C(=O)C3(O)C(O)=C(C(N)=O)C(=O)C(N
(C)C)C3C(O)C21
[02:34:29] New largest fragment: CC1c2cccc(O)c2C(O)=C2C(=O)C3(O)C(O)=C(C
(N)=O)C(=O)C(N(C)C)C3C(O)C21 (56)
[02:34:29] Fragment: Cl
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: [O-]n1ccccc1=S
[02:34:29] New largest fragment: [O-]n1ccccc1=S (12)
[02:34:29] Fragment: [Na+]
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: CN1CCN(CCCN2c3ccccc3Sc3ccc(Cl)cc32)CC1
[02:34:29] New largest fragment: CN1CCN(CCCN2c3ccccc3Sc3ccc(Cl)cc32)CC1 (4
9)
[02:34:29] Fragment: O=C(O)/C=C\C(=O)O
[02:34:29] Fragment: O=C(O)/C=C\C(=O)O
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: CC(CN1c2ccccc2Sc2ccccc21)N(C)C
[02:34:29] New largest fragment: CC(CN1c2ccccc2Sc2ccccc21)N(C)C (40)
[02:34:29] Fragment: Cl
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: Cc1ncc(CO)c(CO)c1O
[02:34:29] New largest fragment: Cc1ncc(CO)c(CO)c1O (23)
[02:34:29] Fragment: Cl
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
```

```
[02:34:29] Running LargestFragmentChooser
[02:34:29] Fragment: CN(C)[C@@H]1C(=O)C(C(N)=O)=C(O)[C@@]2(O)C(=O)C3=C(O)c
4c(O)cccc4[C@@](C)(O)[C@H]3C[C@@H]12
[02:34:29] New largest fragment: CN(C)[C@@H]1C(=O)C(C(N)=O)=C(O)[C@@]2(O)C
(=O)C3=C(O)c4c(O)cccc4[C@@](C)(O)[C@H]3C[C@@H]12 (56)
[02:34:29] Fragment: Cl
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:29] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: CCCCc1oc2ccccc2c1C(=O)c1cc(I)c(OCCN(CC)CC)c(I)c1
[02:34:30] New largest fragment: CCCCc1oc2ccccc2c1C(=O)c1cc(I)c(OCCN(CC)C
C)c(I)c1 (60)
[02:34:30] Fragment: Cl
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: CCCCCCCCCCCC(=O)[O-]
[02:34:30] New largest fragment: CCCCCCCCCCCC(=O)[O-] (37)
[02:34:30] Fragment: [Na+]
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: CCCCCCCCCCCCOS(=O)(=O)[O-]
[02:34:30] New largest fragment: CCCCCCCCCCCCOS(=O)(=O)[O-] (42)
[02:34:30] Fragment: [Na+]
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
```

```
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: CN(C)[C@@H]1C(=O)C(C(N)=O)=C(O)[C@@]2(O)C(=O)C3=C(O)c
4c(O)cccc4[C@@](C)(O)[C@H]3[C@H](O)[C@@H]12
[02:34:30] New largest fragment: CN(C)[C@@H]1C(=O)C(C(N)=O)=C(O)[C@@]2(O)C
(=O)C3=C(O)c4c(O)cccc4[C@@](C)(O)[C@H]3[C@H](O)[C@@H]12 (57)
[02:34:30] Fragment: Cl
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
```

```
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: CCOC(=O)C1=C(COCCN)NC(C)=C(C(=O)OC)C1c1ccccc1Cl
[02:34:30] New largest fragment: CCOC(=O)C1=C(COCCN)NC(C)=C(C(=O)OC)C1c1cc
ccc1Cl (53)
[02:34:30] Fragment: O=S(=O)(O)c1ccccc1
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: COC(=O)C1=C(C)NC(C)=C(C(=O)O[C@@H]2CCCN(Cc3ccccc3)C2)
[C@@H]1c1cccc([N+](=O)[O-])c1
[02:34:30] New largest fragment: COC(=O)C1=C(C)NC(C)=C(C(=O)O[C@@H]2CCCN(C
c3ccccc3)C2)[C@@H]1c1cccc([N+](=O)[O-])c1 (68)
[02:34:30] Fragment: Cl
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: OCCN1CCN(CCCN2c3ccccc3Sc3ccc(C(F)(F)F)cc32)CC1
[02:34:30] New largest fragment: OCCN1CCN(CCCN2c3ccccc3Sc3ccc(C(F)(F)F)cc3
2)CC1 (56)
[02:34:30] Fragment: Cl
[02:34:30] Fragment: Cl
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: COC(=O)C1=C(C)NC(C)=C(C(=O)OCCN(C)Cc2ccccc2)C1c1cccc
([N+](=O)[O-])c1
[02:34:30] New largest fragment: COC(=O)C1=C(C)NC(C)=C(C(=O)OCCN(C)Cc2cccc
c2)C1c1cccc([N+](=O)[O-])c1 (64)
[02:34:30] Fragment: Cl
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Running LargestFragmentChooser
[02:34:30] Fragment: CC(=O)CC(c1ccccc1)c1c([O-])c2ccccc2oc1=O
[02:34:30] New largest fragment: CC(=O)CC(c1ccccc1)c1c([O-])c2ccccc2oc1=O
(38)
[02:34:30] Fragment: [Na+]
[02:34:30] Running LargestFragmentChooser
```

```
Rows: 162/Columns in raw: 230
Columns in X (clean): 191
First rows of X:
 MaxAbsEStateIndex  MaxEStateIndex  MinAbsEStateIndex  MinEStateIndex
qed       SPS   MolWt  HeavyAtomMolWt  ExactMolWt  NumValenceElectrons  Ma
xPartialCharge  MinPartialCharge  MaxAbsPartialCharge  MinAbsPartialCharge
FpDensityMorgan1  FpDensityMorgan2  FpDensityMorgan3  BCUT2D_MWHI  BCUT2D_
MWLOW  BCUT2D_CHGHI  BCUT2D_CHGLO  BCUT2D_LOGPHI  BCUT2D_LOGPLOW  BCUT2D_M
RHI  BCUT2D_MRLOW   AvgIpc  BalabanJ    BertzCT      Chi0     Chi0n     Ch
i0v      Chi1    Chi1n    Chi1v    Chi2n    Chi2v    Chi3n    Chi3v    Chi
4n    Chi4v  HallKierAlpha          Ipc    Kappa1   Kappa2   Kappa3  Labu
teASA  PEOE_VSA1  PEOE_VSA10  PEOE_VSA11  PEOE_VSA12  PEOE_VSA13  PEOE_VSA
14  PEOE_VSA2  PEOE_VSA3  PEOE_VSA4  PEOE_VSA5  PEOE_VSA6  PEOE_VSA7  PEOE
_VSA8  PEOE_VSA9  SMR_VSA1  SMR_VSA10  SMR_VSA3  SMR_VSA4  SMR_VSA5  SMR_V
SA6  SMR_VSA7  SMR_VSA9  SlogP_VSA1  SlogP_VSA10  SlogP_VSA11  SlogP_VSA12
SlogP_VSA2  SlogP_VSA3  SlogP_VSA4  SlogP_VSA5  SlogP_VSA6  SlogP_VSA7  Sl
ogP_VSA8   TPSA  EState_VSA1  EState_VSA10  EState_VSA11  EState_VSA2  ESt
ate_VSA3  EState_VSA4  EState_VSA5  EState_VSA6  EState_VSA7  EState_VSA8
EState_VSA9  VSA_EState1  VSA_EState10  VSA_EState2  VSA_EState3  VSA_ESta
te4  VSA_EState5  VSA_EState6  VSA_EState7  VSA_EState8  VSA_EState9  Frac
tionCSP3  HeavyAtomCount  NHOHCount  NOCount  NumAliphaticCarbocycles  Num
AliphaticHeterocycles  NumAliphaticRings  NumAmideBonds  NumAromaticCarboc
ycles  NumAromaticHeterocycles  NumAromaticRings  NumAtomStereoCenters  Nu
mBridgeheadAtoms  NumHAcceptors  NumHDonors  NumHeteroatoms  NumHeterocycl
es  NumRotatableBonds  NumSaturatedCarbocycles  NumSaturatedHeterocycles
NumSaturatedRings  NumSpiroAtoms  NumUnspecifiedAtomStereoCenters      Phi
RingCount  MolLogP   MolMR  fr_Al_COO  fr_Al_OH  fr_Al_OH_noTert  fr_ArN
fr_Ar_COO  fr_Ar_N  fr_Ar_NH  fr_Ar_OH  fr_COO  fr_COO2  fr_C_O  fr_C_O_no
COO  fr_HOCCN  fr_Imine  fr_NH0  fr_NH1  fr_NH2  fr_N_O  fr_Ndealkylation1
fr_Ndealkylation2  fr_Nhpyrrole  fr_alkyl_halide  fr_allylic_oxid  fr_amid
e  fr_amidine  fr_aniline  fr_aryl_methyl  fr_azo  fr_benzene  fr_bicyclic
fr_dihydropyridine  fr_ester  fr_ether  fr_furan  fr_guanido  fr_halogen
fr_hdrzine  fr_hdrzone  fr_imidazole  fr_imide  fr_ketone  fr_ketone_Topli
ss  fr_lactam  fr_lactone  fr_methoxy  fr_nitro  fr_nitro_arom  fr_nitro_a
rom_nonortho  fr_para_hydroxylation  fr_phenol  fr_phenol_noOrthoHbond  fr
_piperdine  fr_piperzine  fr_priamide  fr_pyridine  fr_sulfide  fr_sulfona
md  fr_sulfone  fr_tetrazole  fr_thiazole  fr_thiophene  fr_unbrch_alkane
fr_urea
         11.151855       11.151855           0.177547       -0.503545 0.87
7602 30.250000 324.424         300.232  324.183778                  126
0.119124         -0.496743             0.496743             0.119124
1.416667          2.291667          3.000000    16.465327      9.733492
2.417618     -2.502700       2.417007       -2.544133     5.834889
-0.044441 2.661658  1.687085 760.387301 16.681434 14.058926 14.058926 11.7
07040 8.682990 8.682990 6.974394 6.974394 5.819484 5.819484 4.444306 4.444
306          -1.91 512499.659037 15.608011 6.187608 2.502406 142.313426
9.843390    5.749512    0.000000         0.0    0.000000    0.000000   9.8
83888   0.000000   0.000000   6.578936   6.076020  61.051132  24.169665  1
8.730465  9.843390  10.902925  9.883888 11.835812 24.987450 20.199310 48.6
80719  5.749512    4.736863          0.0     5.749512          0.0   41.23
1567    0.000000   11.835812   24.509061   43.117268         0.0   10.9029
25  45.59     6.103966      5.106527           0.0     6.041841    11.8358
12    41.726223     6.420822    13.306641    24.265468    22.538844     4.
736863     5.349572           0.0     6.858527    12.130807     1.846194
2.006859     7.957657     5.620375     6.068442     1.661567      0.450000
24          1        4                         0                         3
3              0                       1                       1
2                    5                   2             4           1
4               4                  4                         0
3                 3             0                                1 4.024
```

```
010          5  3.17320 95.0268          0         1                1
0          0        1         0         0       0        0       0
0         0         0       2       0       0       0                   0
0             0                0               0         0           0
0               0       0           1            4                    0
0         1         0          0           0          0           0
0         0          0                  0          0           0
1         0              0                       0                      0
0                       0            3             0           0
1           0             0           0             0           0
0                0        0
         12.268665       12.268665           0.241088       -4.588877 0.66
0374 11.142857 308.311         296.215  308.035459                  110
0.297751         -0.507039             0.507039             0.297751
1.142857          1.761905          2.285714    32.239784     10.022356
2.228520     -2.100377       2.321408       -2.067037     7.855380      0.
103128 2.454894  2.629387 780.611574 15.620956 10.977121 11.793617  9.8415
54 5.825519 7.265583 4.190432 5.580391 2.840472 3.715321 1.837219 2.500497
-2.38  42155.432869 15.017356 5.547507 3.037599 120.916319   9.843390   16
.394507    5.783245         0.0   10.118127    0.000000   9.347287   0.000
000   8.417797   0.000000  30.331835   6.066367  11.629819  12.673249 27.6
08474  15.901372  0.000000  0.000000  4.895483  7.109798 53.591472 11.4990
24    4.736863          0.0    11.499024          0.0   30.970117   10.118
127    0.000000   15.921440   47.360053         0.0    0.000000 100.90
26.546367     18.318862           0.0    16.876415     0.000000    12.1327
34    19.242532    18.199101     0.000000     0.000000     9.289613    36.
586252           0.0    11.687259     9.867079     0.036000    -1.256141
9.902413     0.000000     0.000000    -3.406195      0.071429
21          2        6                        0                         0
0              0                       2                         0
2                    0                  0              5           2
7                0                  4                          0
0                  0              0                                  0 3.967
090          2  1.87850 74.3479          0         0               0
0          0        0         0         1       0        0       1
1         0         0       0       0       0       0                   0
0             0                0               0         0           0
0               0       0           2            0                    0
0         1         0          0           0          0           0
0         0          1                  0          0           0
1         0              0                       0                      0
1                       1             0             0           0
0           0             0           0             0           0
0                0        0
         11.165950       11.165950           0.325325       -0.325325 0.70
5368 10.533333 204.225         192.129  204.078644                   78
0.335962         -0.493745             0.493745             0.335962
1.333333          2.133333          2.866667    16.476922     10.129957
2.026932     -2.042365       2.213955       -1.980500     5.806853      0.
339807 2.167858  2.653374 540.048796 10.836499  8.741253  8.741253  7.2027
09 4.877663 4.877663 3.339426 3.339426 2.233829 2.233829 1.561695 1.561695
-1.77   2829.054631  9.772439 3.768723 1.860292  87.287778   9.154014   11
.332532    0.000000         0.0    0.000000    5.625586   0.000000   4.794
537   0.000000   0.000000   0.000000  31.543660  17.518958   6.606882  9.1
54014  10.969244  0.000000  0.000000 13.847474  6.606882 40.249043  5.7495
12   10.362449          0.0     5.749512          0.0    6.606882    0.000
000    6.923737   12.487189   33.477156         0.0   10.969244  39.44
0.000000      4.794537           0.0     5.625586    12.189902    16.69918
8     6.066367     6.066367    25.980209     0.000000     9.154014    10.4
25547           0.0    11.165950     0.942642     1.170275     0.721065
```

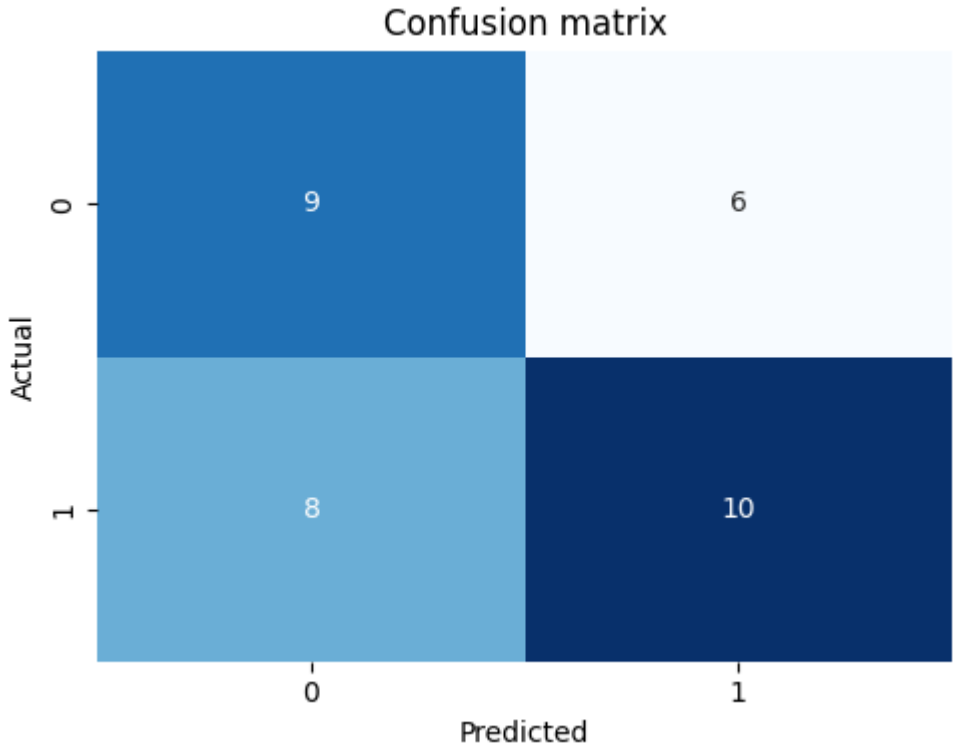
```
7.007289     0.000000     4.400566     0.000000      0.250000
15          0        3                         0                          0
0              0                        1                          1
2                     0                    0              3           0
3                1                   2                           0
0                  0               0                                  0 2.455
308          2  2.50012 58.3900          0         0                 0
0          0        0         0         0       0        0       0
0         0         0       0       0       0       0                   0
0             0                0                0         0          0
0               1       0          1           1                  0
0         1         0           0           0           0          0
0         0          0                   0          0           0
0         0              0                         0                         0
0                       0             0             0            0
0           0             0           0             0            0
0                 0        0
         10.801691       10.801691           0.299673       -0.299673 0.55
2257 10.416667 160.172         152.108  160.052429                   60
0.335690         -0.422729             0.422729             0.335690
1.166667          2.000000          2.833333    16.391868     10.153925
1.946105     -1.951119       2.120754       -1.934028     5.762922      0.
560734 2.053587  2.873950 468.467428  8.552042  6.703248  6.703248  5.7540
20 3.761090 3.761090 2.795711 2.795711 1.774814 1.774814 1.191288 1.191288
-1.57    692.058854  7.099295 2.482514 1.114984  69.444349   4.417151    5
.583020    0.000000         0.0    0.000000    5.625586   0.000000   4.794
537   0.000000   0.000000  11.629819  25.122838  11.452591   0.000000  4.4
17151  10.969244  0.000000  0.000000  6.923737  0.000000 46.315410  0.0000
00    5.625586          0.0     0.000000          0.0    0.000000    0.000
000    6.923737    5.563451   39.543523         0.0   10.969244  30.21
0.000000      4.794537           0.0     5.625586     5.583020    10.94967
6     6.066367     6.066367    25.122838     0.000000     4.417151     4.9
61065           0.0    10.801691     0.967222     1.510002     0.000000
8.922176     0.000000     2.004511     0.000000      0.100000
12          0        2                         0                          0
0              0                        1                          1
2                     0                    0              2           0
2                1                   0                           0
0                  0               0                                  0 1.468
675          2  2.10142 47.2210          0         0                 0
0          0        0         0         0       0        0       0
0         0         0       0       0       0       0                   0
0             0                0                0         0          0
0               1       0          1           1                  0
0         0         0          0           0           0          0
0         0          0                   0          0           0
0         0              0                         0                         0
0                       0             0             0            0
0           0             0           0             0            0
0                 0        0
         10.875576       10.875576           0.346525       -0.346525 0.62
1753 10.230769 176.171         168.107  176.047344                   66
0.335693         -0.496624             0.496624             0.335693
1.230769          2.000000          2.769231    16.466276     10.208834
1.975995     -1.992579       2.179711       -1.938475     5.771611      0.
413783 2.163185  2.746434 484.738995  9.259149  7.111496  7.111496  6.2920
25 3.873462 3.873462 2.658188 2.658188 1.830255 1.830255 1.167956 1.167956
-1.77   1243.831488  7.857386 2.935678 1.402156  74.557894   9.154014   11
.332532    0.000000         0.0    0.000000    5.625586   0.000000   4.794
537   0.000000   0.000000   0.000000  18.199101  17.518958   7.109798  9.1
```

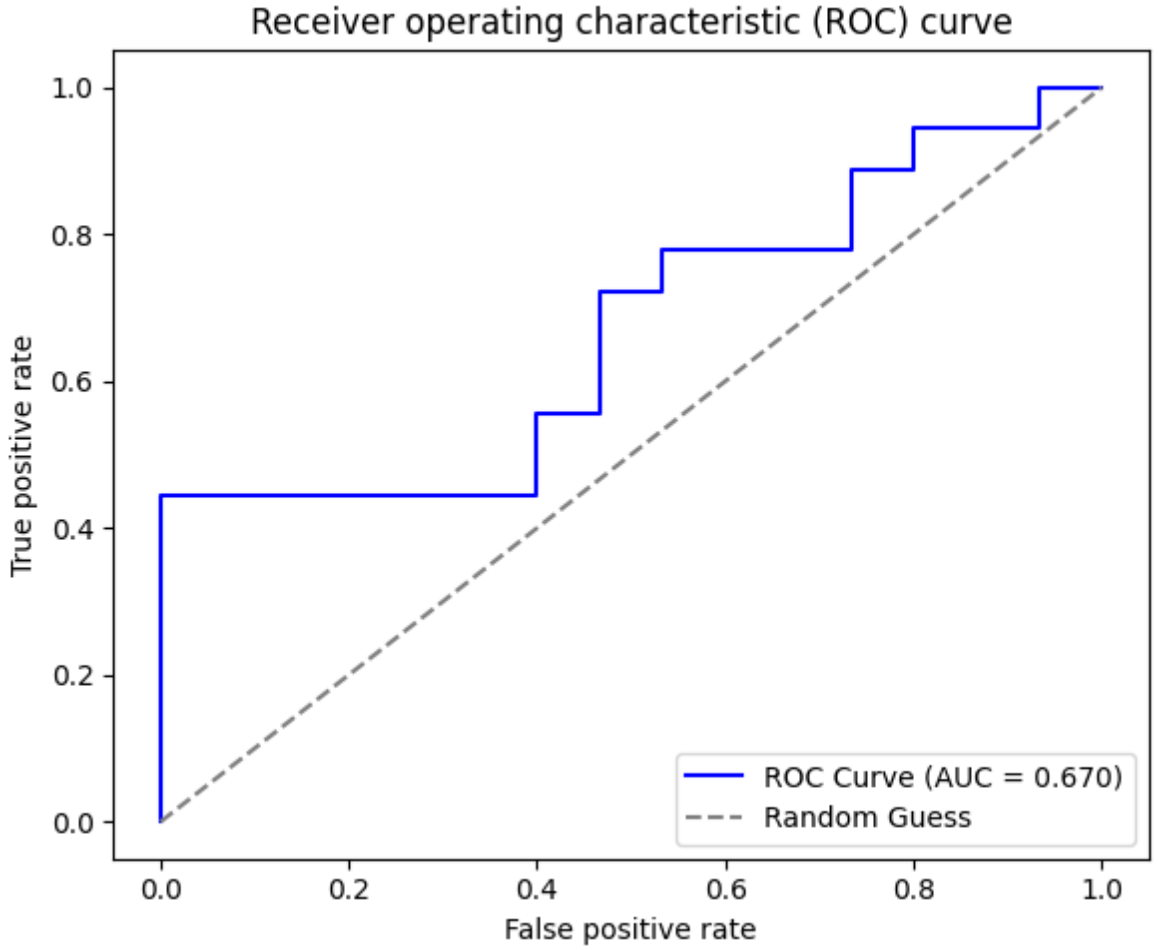
```
54014  10.969244  0.000000  0.000000  0.000000  7.109798 40.751959  5.7495
12   10.362449          0.0     5.749512          0.0    7.109798    0.000
000    0.000000    0.000000   39.543523         0.0   10.969244  39.44
0.000000      4.794537           0.0     5.625586    11.332532     5.38622
4     6.066367    19.242532    12.132734     0.000000     9.154014     9.9
68674           0.0    10.875576     0.889259     0.201484     0.682269
8.477764     0.000000     0.000000     1.571641      0.100000
13          0        3                        0                          0
0              0                       1                        1
2                    0                   0              3           0
3                1                 1                          0
0                  0               0                                0 1.774
366          2  1.80160 49.0360          0         0                 0
0          0        0         0         0       0        0       0
0         0         0       0       0       0       0                  0
0             0                0               0         0           0
0               0       0           1            1                    0
0         1         0           0           0           0           0
0         0          0                   0          0           0
1         0              0                       0                      0
0                        0             0             0            0
0           0             0           0             0            0
0                 0        0
```

In [67]:
```
DESC_CSV = "X_1D2D_descriptors.csv"
ORIG_XLSX = "in_chemico_dataset.xlsx"
TARGET_COL = "Phototoxicity"
EXCEL_SKIPROWS = 1

# Output
TOP_FEATURES_CSV = "top_features_best.csv"

# Split train-test
X = pd.read_csv(DESC_CSV)
y = pd.read_excel(ORIG_XLSX, engine="openpyxl", skiprows=EXCEL_SKIPROWS)[

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Initial model
base_model = xgb.XGBClassifier(
    n_estimators=400,
    learning_rate=0.05,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    n_jobs=-1,
    tree_method="hist",
    eval_metric="logloss",
)
base_model.fit(X_train, y_train)

# Get features
importances = base_model.feature_importances_
feat_imp = pd.DataFrame({
    "feature": X.columns,
    "importance": importances
}).sort_values("importance", ascending=False)
```

```
print("\nTop 10 features:")
print(feat_imp.head(10))

y_pred = base_model.predict(X_test)
y_prob = base_model.predict_proba(X_test)[:, 1]

# Metrics
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_prob)

print("\nModel metrics")
print(f"Accuracy:  {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall:    {rec:.4f}")
print(f"F1 Score:  {f1:.4f}")
print(f"ROC AUC:   {auc:.4f}")

# Classification report
print("\nClassification report:")
print(classification_report(y_test, y_pred, digits=3))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion matrix")
plt.tight_layout()
plt.show()


# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, color="blue", label=f"ROC Curve (AUC = {auc:.3f})")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Random Gues
plt.xlabel("False positive rate")
plt.ylabel("True positive rate")
plt.title("Receiver operating characteristic (ROC) curve")
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()
```

```
Top 10 features:
            feature  importance
58        PEOE_VSA8    0.031478
104         NOCount    0.029348
62         SMR_VSA3    0.029012
108   NumAmideBonds    0.024193
30            Chi0v    0.023395
128       fr_Al_COO    0.017744
78       SlogP_VSA8    0.017709
39            Chi4v    0.016871
1    MaxEStateIndex    0.016812
92     VSA_EState10    0.016808

Model metrics
Accuracy:  0.5758
Precision: 0.6250
Recall:    0.5556
F1 Score:  0.5882
ROC AUC:   0.6704

Classification report:
              precision    recall  f1-score   support

           0      0.529     0.600     0.562        15
           1      0.625     0.556     0.588        18

    accuracy                         0.576        33
   macro avg      0.577     0.578     0.575        33
weighted avg      0.582     0.576     0.577        33
```

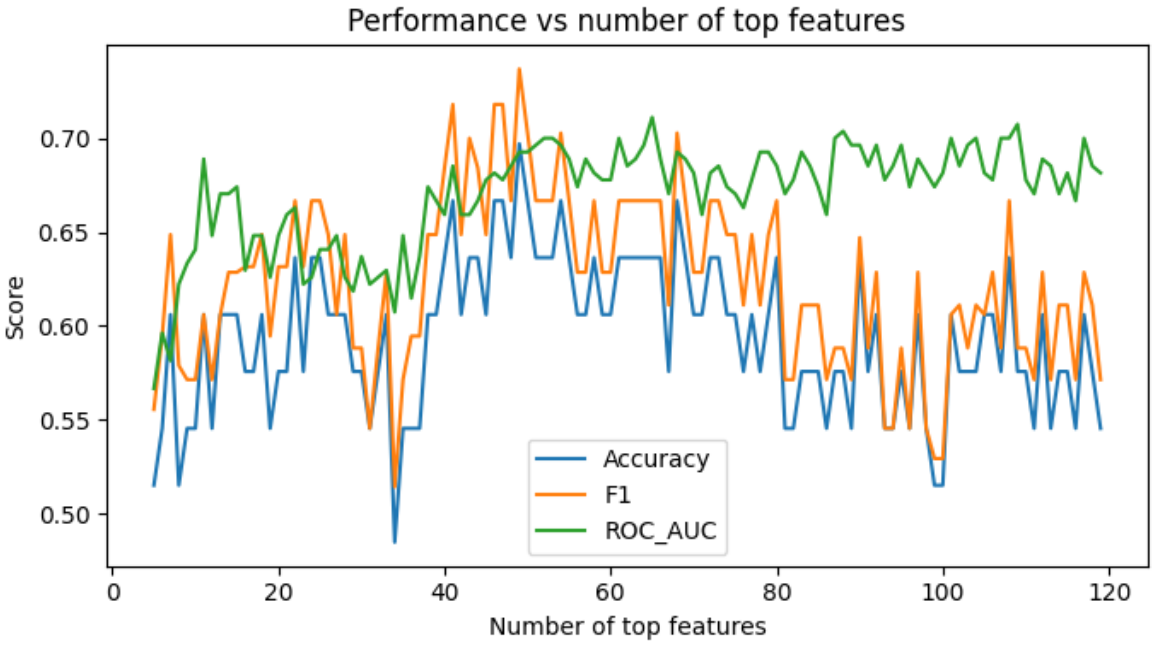




In [68]:
```
results = []

for n in range(5, 120, 1):  # test top 5–119 features
    top_feats = feat_imp["feature"].head(n).tolist()

    model = xgb.XGBClassifier(
        n_estimators=400,
        learning_rate=0.05,
        max_depth=6,
        subsample=0.8,
        colsample_bytree=0.8,
        random_state=42,
        n_jobs=-1,
        tree_method="hist",
        eval_metric="logloss",
    )
    model.fit(X_train[top_feats], y_train)

    y_pred = model.predict(X_test[top_feats])
    y_prob = model.predict_proba(X_test[top_feats])[:, 1]

    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_prob)

    results.append((n, acc, f1, auc))

# Summary
res_df = pd.DataFrame(results, columns=["Top_N", "Accuracy", "F1", "ROC_A
```

```
# Find best by F1 score
best = res_df.iloc[res_df["F1"].idxmax()]
best_n = int(best.Top_N)

# Save the best top features
best_feats = feat_imp.head(best_n)
best_feats.to_csv(TOP_FEATURES_CSV, index=False)
print(f"Saved top {best_n} features -> {TOP_FEATURES_CSV}")

# All in one
plt.figure(figsize=(7,4))
plt.plot(res_df["Top_N"], res_df["Accuracy"], label="Accuracy")
plt.plot(res_df["Top_N"], res_df["F1"], label="F1")
plt.plot(res_df["Top_N"], res_df["ROC_AUC"], label="ROC_AUC")
plt.xlabel("Number of top features")
plt.ylabel("Score")
plt.title("Performance vs number of top features")
plt.legend()
plt.tight_layout()
plt.show()

print("\nBest number of features:", best_n)
print(best)
```

```
Saved top 49 features -> top_features_best.csv
```




```
Best number of features: 49
Top_N       49.000000
Accuracy     0.696970
F1           0.736842
ROC_AUC      0.692593
Name: 44, dtype: float64
```

In [69]:
```
# Best pick
top_feats = feat_imp["feature"].head(best_n).tolist()
final_model = xgb.XGBClassifier(
    n_estimators=400,
    learning_rate=0.05,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
```

```
    n_jobs=-1,
    tree_method="hist",
    eval_metric="logloss",
)
final_model.fit(X_train[top_feats], y_train)

# Predictions for ROC
y_prob_best = final_model.predict_proba(X_test[top_feats])[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob_best)
auc_best = roc_auc_score(y_test, y_prob_best)

# Plot ROC
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, color="blue", label=f"Best Model (Top {best_n} Feats)
plt.plot([0,1],[0,1],"--",color="gray",label="Random Guess")
plt.xlabel("False positive Rate")
plt.ylabel("True positive Rate")
plt.title("ROC curve for best feature count")
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()
```