Report On Project of QASR Aquatic Toxicity (Machine Learning Project using Python)

INT 354
(Advanced Machine Learning)

Name- **Abir Saha**

Reg No- 12009492

Section- KM040

Roll No- RKM040A13

Submitted To- **Dr. Usha Mittal**

Github: https://github.com/abirsaha21/QASR-Aquatic-Toxicity

Abstract

The release of chemicals into aquatic environments has become a major concern due to their potential adverse effects on aquatic organisms. Quantitative StructureActivity Relationship (QSAR) models have emerged as a powerful tool to predict the toxicity of chemicals based on their molecular structure. In this project, QSAR models will be developed for predicting the aquatic toxicity of chemicals. A dataset of chemicals with known toxicity and their corresponding molecular structures will be used to train and validate the models. Different machine learning algorithms will be tested, and the models' performance will be evaluated using statistical metrics such as accuracy, precision, and recall. The models' applicability domains will also be assessed to ensure their reliability and robustness. The developed QSAR models will provide a rapid and cost-effective alternative to experimental testing, allowing for the identification of hazardous chemicals and supporting regulatory decisionmaking. Overall, this project aims to contribute to the development of more sustainable and environmentally friendly chemicals.

Introduction

Quantitative Structure-Activity Relationship (QSAR) is a computational modeling technique used to predict the biological activity of chemical compounds based on their molecular structure. In the case of aquatic toxicity, QSAR models are developed to predict the toxicity of chemicals to aquatic organisms such as fish, algae, and crustaceans.

QSAR models use statistical and machine learning techniques to identify patterns and relationships between the chemical structure of a compound and its toxicity. These models have been developed for a wide range of chemical classes and biological endpoints, including aquatic toxicity. QSAR models for aquatic toxicity can predict the toxicity of chemicals to various aquatic organisms such as fish, algae, and crustaceans.

In this project, QSAR models will be developed to predict the aquatic toxicity of chemicals using a dataset of known toxicity and molecular structures. The models will be trained using various machine learning algorithms and evaluated using statistical metrics such as R-squared and root mean square error. The applicability domains of the models will also be assessed to ensure their reliability and

robustness.The developed QSAR models will provide a rapid and cost-effective alternative to experimental testing, allowing for the identification of hazardous chemicals and supporting regulatory decision-making. Overall, this project aims to contribute to the development of more sustainable and environmentally friendly chemicals.

Dataset Used

For this project, I selected a Kaggle dataset.

Dataset Link:- https://www.kaggle.com/datasets/ishandutta/qsar-fishtoxicitydata-set

Used Libraries

NumPy: A library for numerical computing in Python. It provides functions for working with arrays, matrices, and mathematical operations on them.

Pandas: A library for data manipulation and analysis in Python. It provides data structures and functions for working with tabular data, such as CSV or Excel files.

Matplotlib: A library for creating static, animated, and interactive visualizations in Python. It provides a wide range of charts and graphs for visualizing data.

Seaborn: A library based on Matplotlib that provides a high-level interface for creating statistical graphics. It provides several types of plots such as scatter plots, line plots, bar plots, and heatmaps.

Scikit-learn: A library that provides simple and efficient tools for data mining and data analysis in Python. It includes functions for classification, regression, clustering, and model selection.

Keras: A high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It simplifies the process of building neural networks by providing pre-built layers and models.

TensorFlow: An open-source software library for dataflow and differentiable programming across a range of tasks. It is primarily used for building and training machine learning models, especially deep neural networks.

cv2: A library for computer vision and machine learning in Python. It is an OpenCV library for image and video processing.

Architecture

Data loading: The code starts by loading the input data into a Pandas DataFrame using the pd.read_csv function.

Data preprocessing: The next step involves preprocessing the data by splitting it into the features (X) and target (y) variables, handling missing values in the dataset, and encoding categorical variables using One-Hot Encoding. This is done using Scikit-learn's ColumnTransformer class and SimpleImputer class.

Model training: The decision tree model is built using Scikit-learn's DecisionTreeRegressor class. The hyperparameters for the decision tree model such as maximum depth, minimum samples leaf, and others are set using the class constructor.

Model evaluation: The model's performance is evaluated on a holdout set using the mean squared error metric. This is done using Scikit-learn's mean_squared_error function.

Model deployment: Once the model is trained and evaluated, it can be deployed to make predictions on new data. In this code, predictions are made on the holdout set, and the predicted values are compared with the actual values to calculate the mean squared error.

Pipeline: All the above steps are organized and automated using Scikit-learn's Pipeline class. The pipeline allows chaining the preprocessing steps, model training, and evaluation into a single object, making the code more modular and easier to understand.

Results and Experimental analysis

Data splitting: The dataset is split into training and test sets using the train_test_split function from Scikit-learn. The training set is used for building the model, while the test set is used for evaluating its performance.

Cross-validation: The code includes an optional step for performing k-fold crossvalidation on the training set. Cross-validation is a technique used for evaluating the model's performance on multiple splits of the data. In this code, the cross_val_score function from Scikit-learn is used to perform cross-validation and compute the mean and standard deviation of the scores.

Model hyperparameter tuning: The code includes an optional step for tuning the hyperparameters of the decision tree model. Hyperparameters are parameters that are not learned from the data but are set by the user. In this code, the GridSearchCV function from Scikit-learn is used to search for the best combination of hyperparameters by exhaustively testing all possible combinations.

Model performance evaluation: The code evaluates the performance of the model on the test set using the mean squared error (MSE) metric. The MSE measures the average squared difference between the predicted and actual values, with lower values indicating better performance.

Results analysis: The code prints out the MSE scores obtained during crossvalidation and on the test set. It also generates a scatter plot of the predicted and actual values for the test set, which provides a visual representation of the model's performance. check it by its summary, data shape and a visualization.

Once I ready with the final model, I compile and train the model. Here, the model will be trained in 50 epochs only.

All output Screenshots

```
import random
import seaborn as sns
import matplotlib.pyplot as plt
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/qsar-fish-toxicity-data-set/qsar_fish_toxicity.csv
```

[2]:

```
df = pd.read_csv("/kaggle/input/qsar-fish-toxicity-data-set/qsar_fish_toxicity.csv",header=None,delimiter=';')
df.head()
```

[2]:

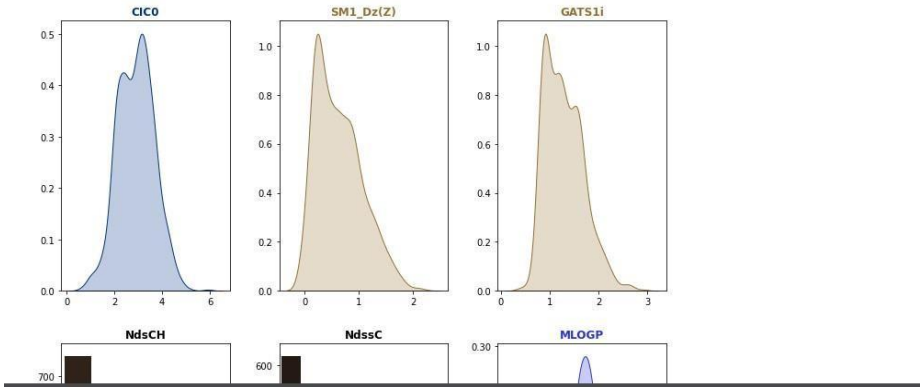
	0	1	2	3	4	5	6
0	3.260	0.829	1.676	0	1	1.453	3.770
1	2.189	0.580	0.863	0	0	1.348	3.115
2	2.125	0.638	0.831	0	0	1.348	3.531
3	3.027	0.331	1.472	1	0	1.807	3.510
4	2.094	0.827	0.860	0	0	1.886	5.390

```
useful_cols = [col for col in df.columns if col not in ['quantitative response, LC50 [-LOG(mol/L)]']]
# TARGET AND ID
cols_dist = [col for col in useful_cols if col not in ['NdssC', 'NdsCH']]  # CATEGORICAL DATA
color_ = [ '#9D2417', '#AF41B4', '#003389' ,'#3C5F41',  '#967032', '#2734DE']
cmap_ = ['magma', 'copper', 'crest']


plt.figure(figsize= (10,10))
for i,col in enumerate(df[useful_cols].columns):
    rand_col = color_[random.sample(range(6), 1)[0]]
    plt.subplot(2,3, i+1) # SUBPLOTS
    if col in cols_dist:

        sns.kdeplot(df[col], color = rand_col, fill = rand_col )
        plt.title(col,weight = 'bold', color = rand_col)
        plt.ylabel(" ")
        plt.xlabel(" ")
```

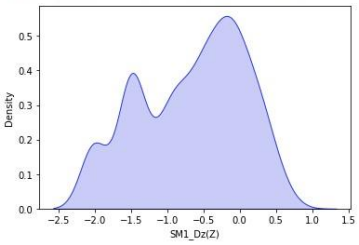
```
plt.show();
```



[8]:
```
sns.kdeplot(np.log(df['SM1_Dz(Z)']), color = rand_col, fill = rand_col )
```

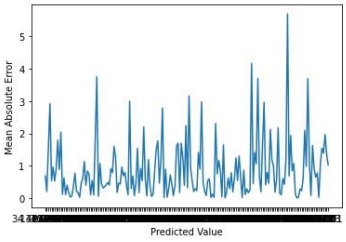
```
/opt/conda/lib/python3.7/site-packages/pandas/core/arraylike.py:364: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

[8]: `<AxesSubplot:xlabel='SM1_Dz(Z)', ylabel='Density'>`



+ Code + Markdown

```
pyplot.plot(errors)
pyplot.xticks(ticks=[i for i in range(len(errors))], labels=predicted)
pyplot.xlabel('Predicted Value')
pyplot.ylabel('Mean Absolute Error')
pyplot.show()
```



+ Code + Markdown

Conclusion and Future Scope

Google has improved its original translation software significantly using artificial intelligence. And translation companies will integrate similar software into their workflows, to deliver precise translations faster, for broader audiences. At the moment, machine translation (MT) still has difficulties with translating into many of the 7,000 languages and dialects. Mastering artificial intelligence and deep learning will create a new generation of the translation software. One that delivers more accurate versions of the original content, in more languages. The future of translation will cover more cultures, as the internet continues to penetrate emerging countries worldwide. Besides the top languages for translation, the software will have to provide accurate solutions to communicate with audiences who speak less known dialects.

The software of the future is easy to use, has a minimal design, and can connect in seconds with complementary technology. Translators won't need IT specialists to provide consultancy, as the software will have high usability and an intuitive interface.

As machines improve, translators will have to adapt their skills to deliver customized services, such as a deep understanding of the target public, which allows them to adjust the translated content for segmented audiences.

References

https://www.oecd.org/chemicalsafety/risk-assessment/theoecdqsartoolbox.htm

https://comptox.epa.gov/dashboard

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3644977/

https://pubs.acs.org/doi/10.1021/acs.est.6b00841

https://www.sciencedirect.com/science/article/pii/S016041201400114X