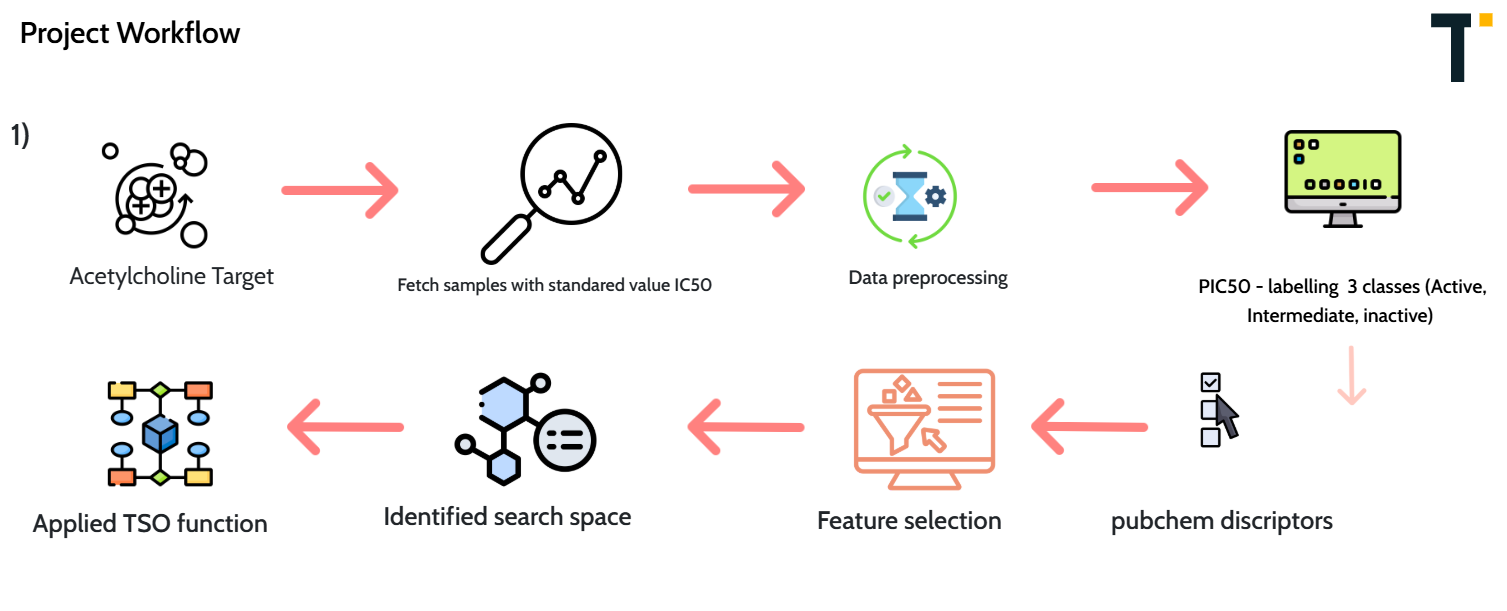
## **Project Documentation**

### **1. Project Architecture**



1. **Data Preprocessing**:  
   * Three datasets were created:
     + **Downsampled Data**: Reduced majority class to match minority class size.
     + **Unbalanced Data**: Original dataset used as it is.
     + **Oversampled Data**: Synthetic samples Oversampling Technique (SMOTE) was applied to balance the dataset.
   * Data splitting: 80% training and 20% testing.
2. **Model Building**:  
   * Machine Learning Models:
     + Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbors (KNN), stacking, XGBoost (XGB).
   * Models were trained and evaluated on each dataset (downsampled, unbalanced, and oversampled).
3. **Optimization**:  
   * **Tuna Swarm Optimization (TSO)** was used to fine-tune the hyperparameters for each model.
   * Fitness function: Cross-validation accuracy.
4. **Output**:  
   * Best-performing models with optimized hyperparameters for each dataset.
   * Benchmarking for the models with the downsampling, unbalanced, and oversampling

### **2. Design Decisions**

1. **Data Handling**:

* **Imbalance Issue**: Addressed using downsampling and oversampling techniques to ensure fair performance evaluation.
* Choice of SMOTE for oversampling was made due to its ability to create synthetic samples, enhancing generalization.

1. **Feature selection:**
   * Using recursive feature elimination with cross validation with random forest as an estimator
2. **Model Selection**:
   * Diverse models were chosen to test performance across complexity levels (e.g., simpler models like DT and KNN vs. advanced models like XGBoost and stacking).
3. **Optimization Algorithm**:  
   * **TSO** was selected for hyperparameter optimization due to its proven efficiency in handling large search spaces compared to grid or random search.
4. **Evaluation Metrics**:  
   * Chosen metrics were designed to evaluate performance on imbalanced datasets, prioritizing recall and F1-score for the minority class.

### **3. Algorithms Used**

1. **Tuna Swarm Optimization (TSO)**:  
   * Inspired by the foraging and migration behavior of tuna fish.
   * The algorithm maintains a balance between exploration (searching new areas) and exploitation (refining existing solutions).
   * **Steps**:
     + Initialize a population of candidate solutions.
     + Evaluate fitness function for each candidate (cross-validation accuracy).
     + Update positions of candidates based on leader-following and random migration.
     + Iterate until convergence or a maximum number of iterations is reached.
2. **Synthetic Minority Oversampling Technique (SMOTE)**:  
   * Generates synthetic examples for the minority class by interpolating between existing minority class samples and their nearest neighbors.
   * Reduces overfitting while addressing class imbalance.
3. **Machine Learning Models**:  
   * **Random Forest (RF)**: Ensemble method using decision trees with bagging.
   * **Support Vector Machine (SVM)**: Maximizes margin between classes using kernel functions.
   * **Decision Tree (DT)**: Constructs a tree based on feature splits to classify data.
   * **K-Nearest Neighbors (KNN)**: Classifies based on the majority label of k nearest data points.
   * **Stacking:** Predict based on the integration of different base models for more accurate predictions.
   * **XGBoost (XGB)**: Gradient boosting framework optimizing model accuracy.

### **4. Dependencies**

#### **Programming Language:**

* **Python 3.9**

#### **Libraries and Frameworks:**

* **Data Handling**:  
  + pandas: For data manipulation.
  + numpy: For numerical operations.
* **Machine Learning**:
* scikit-learn:
  + StackingClassifier: For combining multiple base models using meta-models.
  + LogisticRegression: Used as a meta-classifier for stacked models or as a standalone model.
  + SVM: For implementing Support Vector Machine (SVM) models.
  + RandomForestClassifier, DecisionTreeClassifier, KNeighborsClassifier: For individual machine learning models.
  + SMOTE: For oversampling to address class imbalance.
  + train\_test\_split,
  + XGboost: For the XGBoost model.
* **Optimization**:

Custom implementation of **Tuna Swarm Optimization (TSO)**.

#### **Environment:**

* Development environment: Jupyter Notebook or VS Code.
* Hardware: System with at least 8GB RAM and a multi-core processor for faster computations.