Drug Discovery using AI with OpenVINO and RedHat OpenShift

Steps we have performed

**Step 1: Lipophilicity Prediction**

* **Objective**: Assess the lipophilicity (hydrophobic or hydrophilic characteristics) of drug-like molecules, as it affects the molecule's pharmacokinetics, including absorption, distribution, and solubility.
* **Procedure**:
  + **Data Preparation**: Use a molecular dataset with known lipophilicity values. Ensure molecules are represented using SMILES notation.
  + **Modeling**: Deploy a pre-trained OpenVINO model optimized for regression tasks, fine-tuned to predict lipophilicity.
  + **Deployment on OpenShift**: Containerize the model and deploy it on RedHat OpenShift for scalable predictions. The platform supports container orchestration, making it ideal for real-time lipophilicity assessments.
  + **Output**: Receive predicted lipophilicity scores, allowing you to screen molecules based on their potential suitability in drug development.

**Step 2: Binding Affinity Prediction**

* **Objective**: Evaluate the binding affinity between the molecule and a biological target, which is crucial for efficacy.
* **Procedure**:
  + **Data Preparation**: Gather a dataset with labeled binding affinities for various protein-ligand pairs. Represent ligands in SMILES and proteins in appropriate 3D formats.
  + **Modeling**: Use OpenVINO for accelerated processing and inference, with a focus on predicting binding affinity. Machine learning models like deep docking networks can predict how well the molecule binds to a target protein.
  + **Deployment**: Deploy the affinity model on OpenShift to handle large batches, scaling for drug library screening.
  + **Output**: Generate binding affinity scores, ranking molecules based on their likelihood of binding effectively to the target.

**Step 3: Molecule Generation and Scoring**

* **Objective**: Generate new molecules based on an initial molecular scaffold and score them for potential efficacy.
* **Procedure**:
  + **Scaffold Selection**: Define an initial molecule or scaffold with promising characteristics.
  + **Molecule Generation**: Using RDKit in conjunction with OpenVINO, create variations of the scaffold. RDKit can generate new molecules by modifying functional groups, and OpenVINO accelerates this process with optimized inference.
  + **Scoring**: Score generated molecules on metrics like lipophilicity, binding affinity, and other pharmacokinetic properties using previously deployed models.
  + **Deployment**: The generation and scoring workflow can be containerized and deployed on OpenShift for efficient, parallel processing of molecular variants.
  + **Output**: Rank and select the top candidates with the highest scores for further analysis.

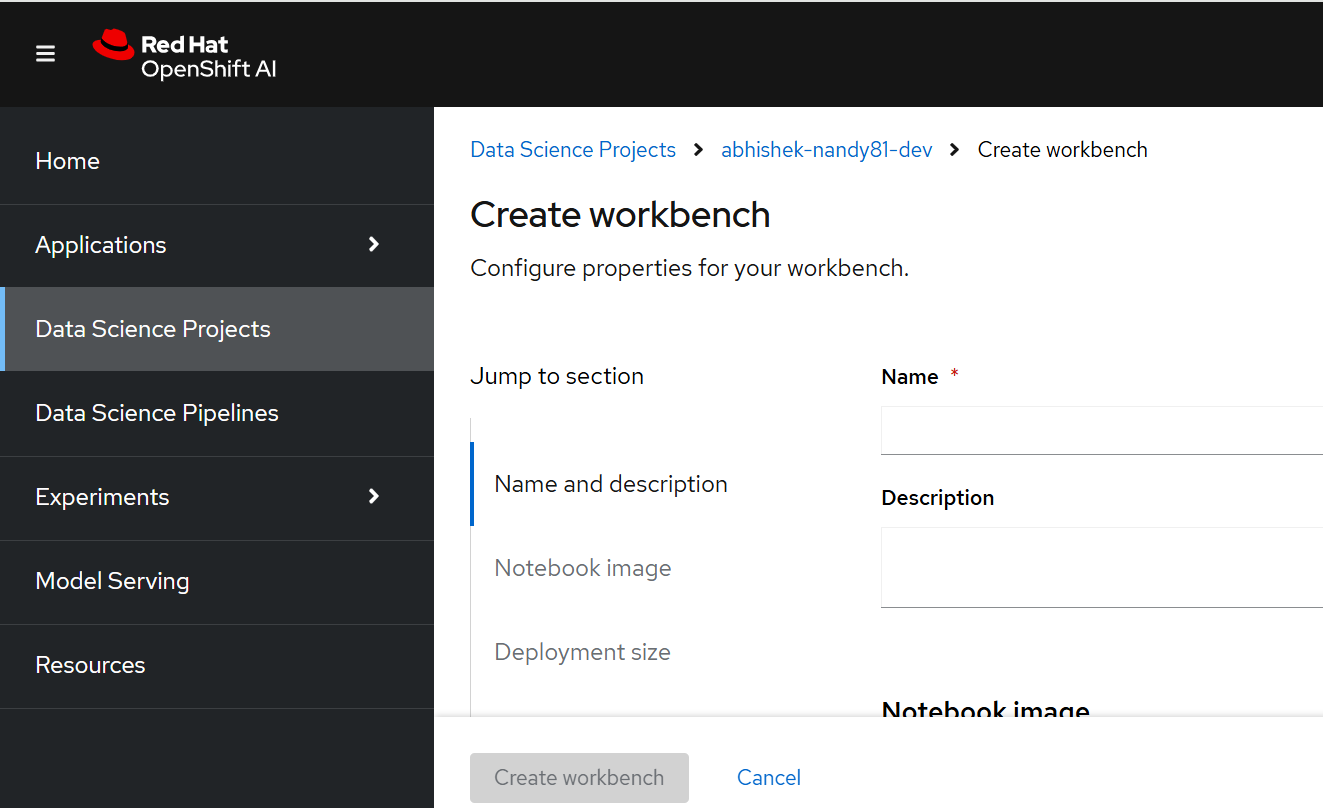
**Step 4: Reaction Prediction and SMILES Visualization**

* **Objective**: Predict chemical reactions and visualize the resulting products to assess synthetic feasibility and optimize the drug synthesis pathway.
* **Procedure**:
  + **Reaction Prediction**: Use OpenVINO-accelerated models trained for reaction prediction. Provide the SMILES representation of reactants, and the model predicts potential products based on known reaction patterns.
  + **Visualization**: Use OpenVINO's capabilities to generate visual representations of SMILES, converting them into a format that chemists can analyze. This aids in understanding the reaction outcomes visually.
  + **Deployment**: Reaction prediction and visualization workflows are deployed on OpenShift to facilitate real-time feedback for synthesis planning.
  + **Output**: Predicted products and their visual representations, assisting in selecting feasible reactions for synthesizing new drug compounds.

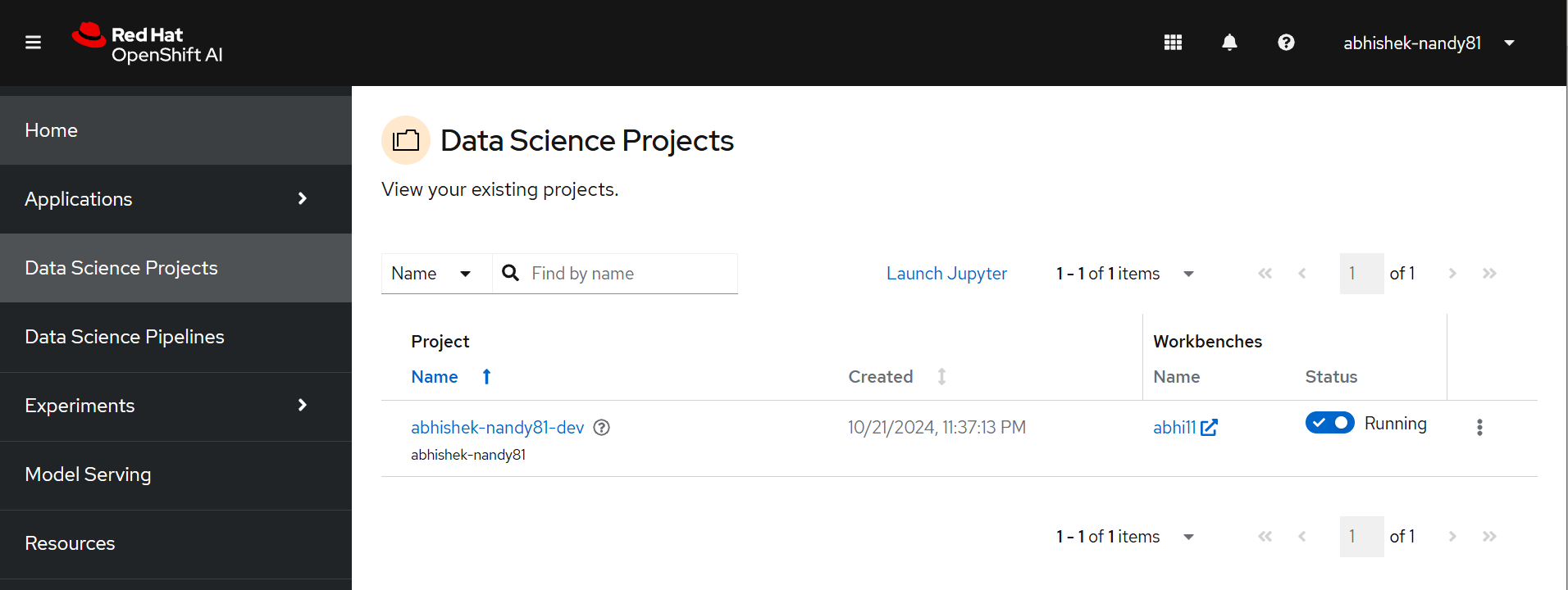
Step 5: Reaction Prediction with RAG (Retrieval-Augmented Generation) Integration

* **Objective**:
  + The goal of integrating Retrieval-Augmented Generation (RAG) is to enhance the accuracy of reaction predictions by combining external data retrieval with generative modeling. By retrieving examples of similar reactions, RAG provides context from historical data, helping to improve prediction quality and relevance.
  + This approach assists researchers by delivering predictions enriched with relevant chemical pathways, thereby informing decisions with contextually accurate and feasible synthetic pathways.
* **Procedure**:
  + Data Retrieval:
    - Objective: Retrieve similar reactions and synthesis pathways from a database to provide historical context for the current prediction.
    - Method: Qdrant is used as a vector database to store reaction data in the form of molecular fingerprints. Each reactant’s structure is represented as a 4096-dimensional vector created using RDKit’s GetMorganGenerator. This vector encoding captures unique structural features of molecules, making it suitable for similarity-based retrieval.
    - Implementation: When a new reaction prediction is initiated, the reactant’s fingerprint vector is used to query Qdrant. This query returns the most similar reactions from the database based on cosine similarity. By pulling these similar reactions, RAG enriches the prediction with relevant examples, giving the model access to pathways and reaction patterns that may align with the input molecules.
  + **Reaction Prediction**:
    - Objective: Generate a prediction for the reaction’s likelihood using a model optimized with OpenVINO, incorporating contextual knowledge from the retrieved data.
    - Model Optimization: The reaction prediction model, trained for feasibility scoring, is converted to OpenVINO format. OpenVINO accelerates inference, allowing the model to generate predictions in real time with CPU optimization.
    - Procedure:
      * Input Preparation: SMILES representations of the reactant and product are converted into 4096-dimensional fingerprint vectors. These vectors are concatenated, forming a single input vector of size 8192, representing both molecules.
      * Inference: OpenVINO performs inference on the combined vector, producing a likelihood score that reflects the model’s confidence in the reaction’s feasibility. This score is augmented with historical examples retrieved from Qdrant, thus benefiting from context-driven insights provided by RAG.
  + **Deployment on OpenShift**:
    - Objective: Deploy the RAG-enabled reaction prediction system on RedHat OpenShift, enabling scalable and efficient processing.
    - Advantages:
      * Scalability: OpenShift container orchestration supports large-scale retrieval and generation workflows, allowing multiple prediction requests to be handled concurrently.
      * Real-time Access: Deployment on OpenShift ensures that predictions and retrievals are processed quickly and can be accessed remotely via Gradio’s web interface, making it suitable for real-time synthesis planning.
    - Implementation: The entire workflow (data retrieval with Qdrant, reaction prediction with OpenVINO, and Gradio interface) is containerized and deployed on OpenShift. This setup facilitates resource scaling based on demand, making it ideal for large datasets or high-throughput screening applications.
  + Output:
    - Enhanced Reaction Predictions: The integration of RAG with OpenVINO generates reaction predictions that are contextually relevant and informed by historical reaction data.
    - Synthesized Pathways: Alongside the predicted likelihood score, similar reactions from the Qdrant database are displayed, giving researchers a synthesized view of feasible reaction pathways. This output supports researchers in selecting pathways that are not only theoretically possible but also practically informed by real-world data.

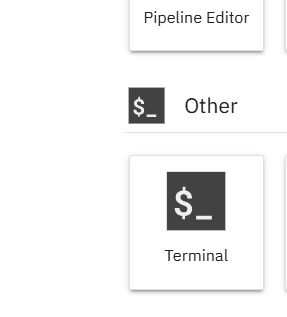
1)Enter the Sandboxed environment.Create a workbench.I have created a small Pytorch one.



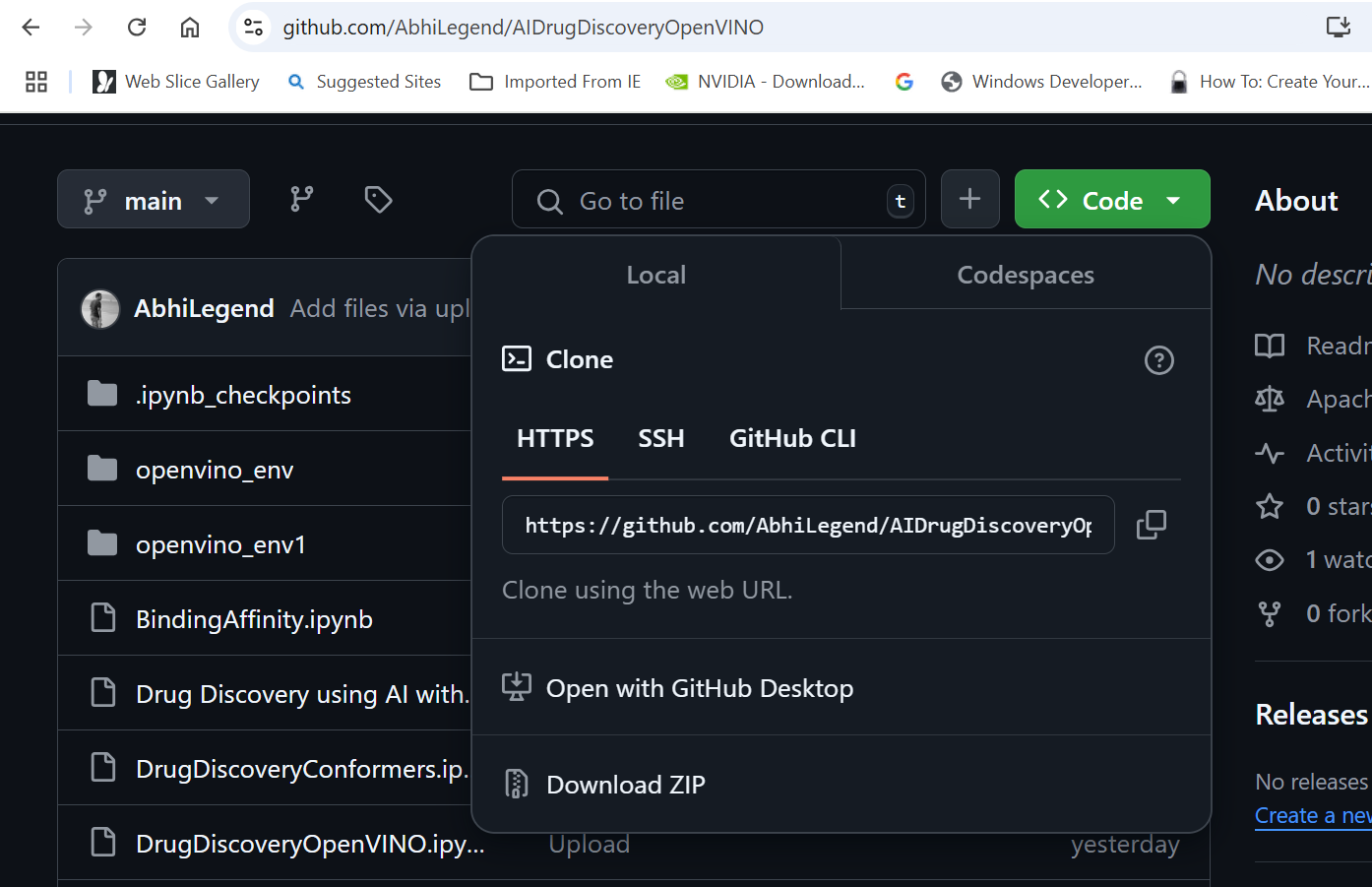
Then open the workbench.



Go to the terminal



Git clone the repo



Git clone https://github.com/AbhiLegend/AIDrugDiscoveryOpenVINO.git

Then create the environment

**Creation of the environment**

Creating a new environment for OpenVINO

Steps

(app-root) (app-root) python3 -m venv openvino\_env1

(app-root) (app-root) source openvino\_env1/bin/activate

(app-root) (app-root) python -m pip install --upgrade pip

Installing ipex Torch

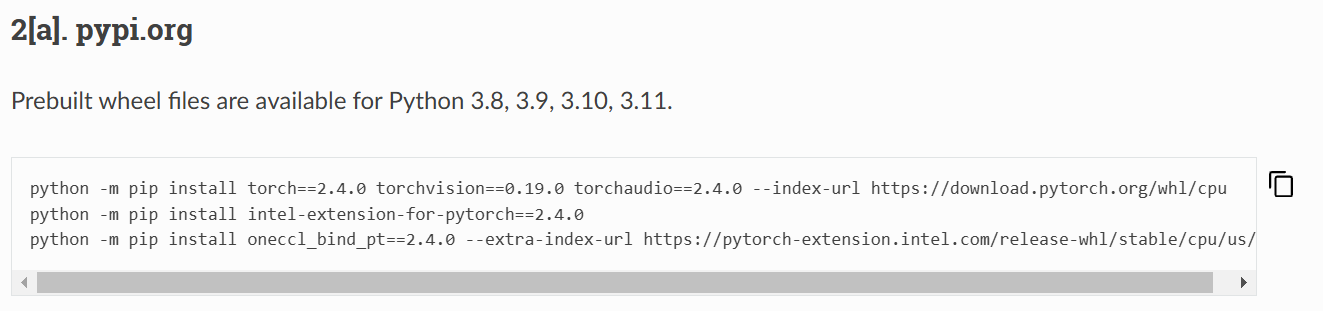
python -m pip install torch==2.4.0 torchvision==0.19.0 torchaudio==2.4.0 --index-url https://download.pytorch.org/whl/cpu

python -m pip install intel-extension-for-pytorch==2.4.0

python -m pip install oneccl\_bind\_pt==2.4.0 --extra-index-url https://pytorch-extension.intel.com/release-whl/stable/cpu/us/

I followed the following document

<https://intel.github.io/intel-extension-for-pytorch/index.html#installation?platform=cpu&version=v2.4.0%2bcpu&os=linux%2fwsl2&package=pip>



OpenVINO installation

(app-root) (app-root) pip install openvino==2024.4.0

Installing RDKit

pip install rdkit

Installing Gradio

pip install gradio

pip install triton

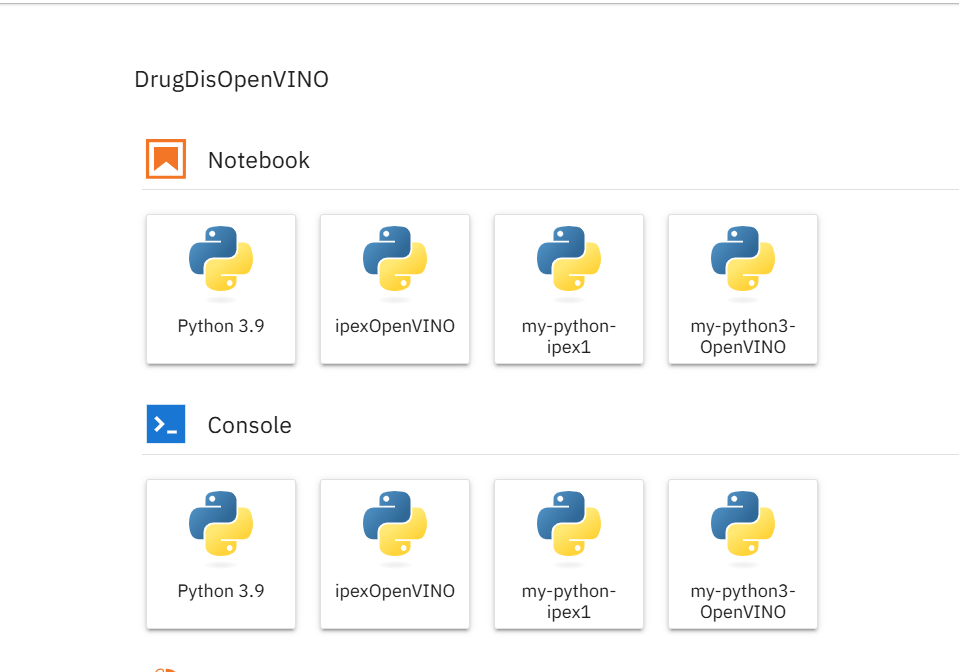
Creating the new kernel

(app-root) (app-root) pip install ipykernel

Register the Kernel

python -m ipykernel install --user --name**=**my-python-ipex1

Now you will be able to see the new kernel in launcher.



**LiphoGradio.ipynb**

Step 1:

What is the notebook doing

In this notebook, the capabilities revolve around predicting molecular lipophilicity values (logD7.4) based on SMILES molecular representations, with enhancements from IPEX and OpenVINO for optimized performance and a Gradio interface for an interactive user experience. Here’s how these components are utilized and the role each one plays:

**Capabilities in the Notebook**

1. **Molecular Fingerprint Conversion**: The notebook uses RDKit to convert SMILES representations into Morgan fingerprints, creating feature vectors that can be fed into the neural network.
2. **Neural Network Model**: A simple three-layer neural network is implemented with PyTorch, specifically structured to predict logD7.4 based on the 2048-bit Morgan fingerprint.
3. **Training Optimization with IPEX**: Intel Extension for PyTorch (IPEX) is used to accelerate model training on Intel CPUs, making the training process more efficient. This is particularly useful when dealing with large datasets or multiple training iterations, as IPEX optimizes both model and optimizer, taking advantage of Intel hardware’s parallel processing capabilities.
4. **Efficient Inference with OpenVINO**: Post-training, the model can be converted and deployed with OpenVINO, which optimizes inference by lowering latency and boosting performance on Intel hardware (including CPUs, GPUs, and VPUs). This approach allows for high-speed predictions, even when screening large numbers of molecular candidates.

**Gradio Frontend Integration**

Gradio provides an interactive web interface, allowing users to input SMILES strings and obtain real-time predictions of logD7.4 values. Here’s how it’s typically set up and enhances the user experience:

1. **User-Friendly Input for SMILES Strings**: Gradio’s interface can prompt users to input a SMILES string, which is then converted to a Morgan fingerprint in the backend.
2. **Real-Time Predictions with OpenVINO**: Once trained, the model optimized by OpenVINO provides quick inference through the Gradio interface. Users receive predictions almost instantly, which is essential in applications like drug discovery where many compounds need to be screened efficiently.
3. **Visualization**: The notebook can use Gradio to display the molecular structure of input SMILES, using RDKit to generate images that make the chemical composition visually interpretable to the user.
4. **Batch Predictions**: Gradio can handle batch processing by accepting multiple SMILES inputs, processing them sequentially or in parallel, and displaying results quickly, which is particularly valuable for high-throughput virtual screening.

**Detailed Workflow Using OpenVINO, IPEX, and Gradio**

1. **Data Processing and Training with IPEX**:
   * IPEX accelerates the training phase by optimizing matrix operations and memory management. This enhancement allows the model to learn from the training data faster and enables more iterations in a shorter time frame.
   * In a Generative AI context, IPEX could also be used to train generative models (e.g., VAEs or GANs) for molecule generation, shortening the training duration and making it feasible to test various configurations efficiently.
2. **Inference Optimization with OpenVINO**:
   * Once trained, the PyTorch model is converted into an OpenVINO-optimized format for inference. OpenVINO minimizes latency, which allows the Gradio frontend to deliver near-instantaneous predictions.
   * OpenVINO’s compatibility with edge devices and cloud environments makes it versatile, enabling deployment in various research and industrial settings.
3. **Gradio for Real-Time Interaction**:
   * Gradio allows researchers and end-users to interact with the model seamlessly, facilitating quick experimentation with different molecular structures.
   * Users can input single or multiple SMILES strings, view molecular structures, and obtain logD7.4 predictions within seconds, creating an engaging, efficient, and accessible platform for research and development.

**Summary**

This notebook combines predictive modeling, Intel-optimized training (IPEX), high-speed inference (OpenVINO), and interactive user engagement (Gradio). Together, these tools create an end-to-end pipeline that is well-suited for molecular property prediction, real-time molecular screening, and Generative AI-based molecular design, advancing applications in drug discovery, material science, and chemistry research.

Testing interface on Gradio

Test Inputs for interface (you can toggle between IPEX and OpenVINO)

Glucose:

SMILES: C(C1C(C(C(C(O1)O)O)O)O)O

A highly polar molecule with very low lipophilicity.

Aspirin:

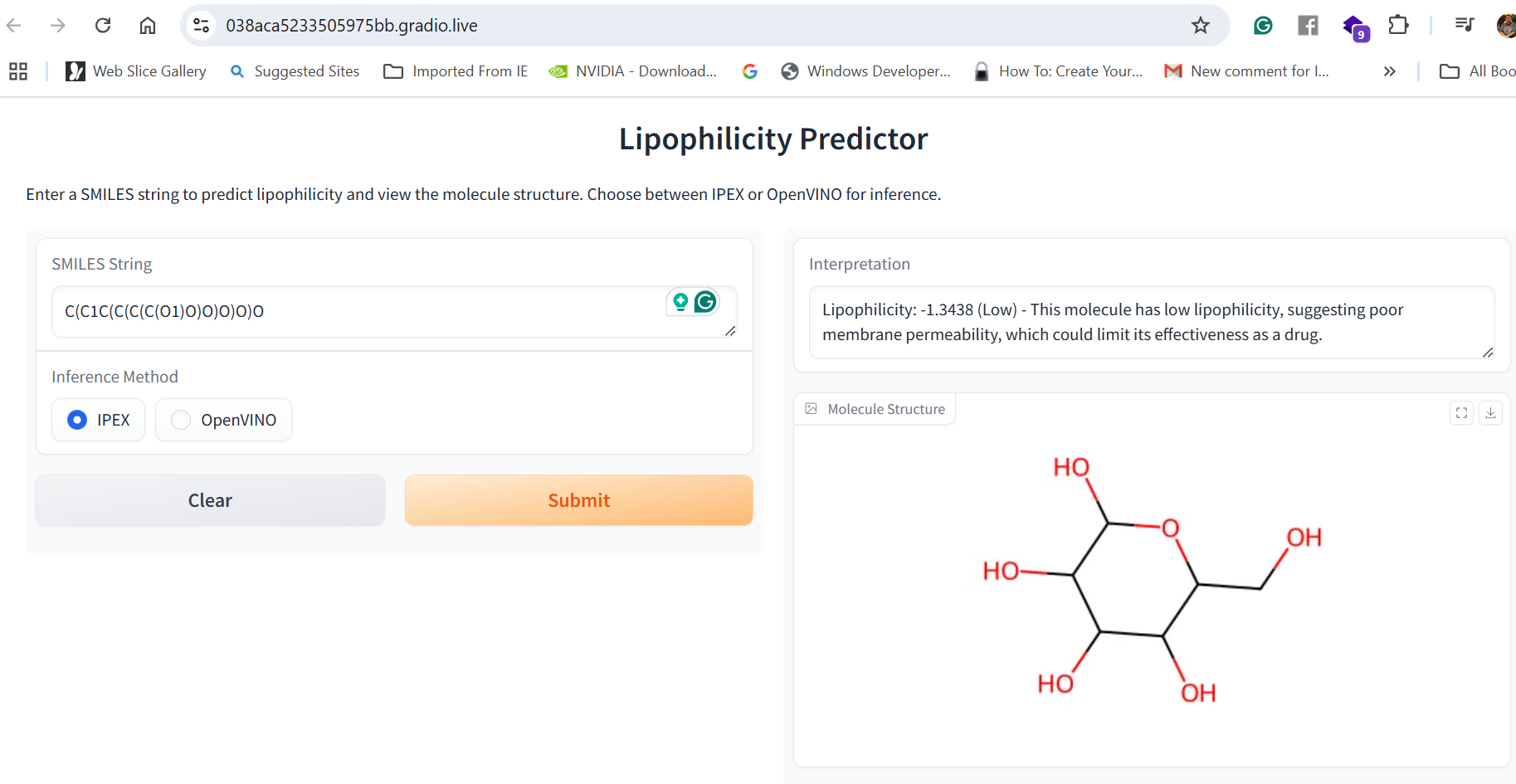
SMILES: CC(=O)OC1=CC=CC=C1C(=O)O

Known for its anti-inflammatory properties, often has moderate lipophilicity.

Caffeine:

SMILES: CN1C=NC2=C1C(=O)N(C(=O)N2C)C

An alkaloid found in coffee, typically has low to moderate lipophilicity.



Step 2:

**BindingAffinity.ipynb**

The notebook focus on predicting binding affinity using a neural network, with optimizations for both training and inference. Here’s a breakdown of how it utilizes Intel Extension for PyTorch (IPEX) and OpenVINO:

**1. Model Training with IPEX**

* **Data Preparation**: A synthetic dataset is created with molecular features (2048-dimensional vectors) and corresponding binding affinity values.
* **Model Definition**: A simple neural network, BindingAffinityModel, is defined with fully connected layers.
* **IPEX Optimization**:
  + The model is scripted using torch.jit.script to prepare it for IPEX optimization.
  + IPEX optimizes the model and optimizer together, enhancing training speed by taking advantage of Intel CPU features.
* **Training Loop**: A loop iterates over the dataset, training the model with Mean Squared Error (MSE) as the loss function. IPEX optimizations likely speed up matrix operations and leverage mixed precision where applicable, providing efficient training on Intel hardware.
* **Model Export**: After training, the model is saved in ONNX format with IPEX optimizations applied. This allows the model to be used by OpenVINO for inference.

**2. Model Inference with OpenVINO**

* **Model Loading and Compilation**:
  + The ONNX model saved during training is loaded into OpenVINO.
  + OpenVINO’s Core API reads and compiles the model for CPU inference, converting it into the Intermediate Representation (IR) format, which is optimized for efficient inference.
* **Saving in IR Format**: The compiled model is saved as an XML file (binding\_affinity\_model\_openvino.xml) with a binary weights file, making it ready for deployment on compatible devices.
* **Inference**:
  + A sample input (2048-dimensional vector) is prepared to test the model.
  + The model runs an inference on this input, and the predicted binding affinity is displayed.
  + OpenVINO’s optimizations provide low-latency, high-throughput inference, ideal for rapid binding affinity predictions in production.

**Summary**

* **Training Phase**: IPEX accelerates training by optimizing the PyTorch model and making efficient use of Intel hardware capabilities.
* **Inference Phase**: OpenVINO takes the optimized ONNX model, compiles it for CPU inference, and runs predictions, achieving fast and efficient inference suitable for large-scale deployment. ​

Step3:

DrugDiscoveryConformers.ipynb

The code in the notebook builds a **drug discovery pipeline** that leverages **Generative AI (GenAI)**, **OpenVINO**, and **IPEX** in the following ways:

**1. Generative AI (GenAI) with VAE**

* A **Variational Autoencoder (VAE)** model is implemented as a generative component to create new molecular fingerprints based on an initial SMILES string scaffold. This GenAI approach uses the latent space of the VAE to sample and generate novel molecular structures that resemble the input structure.
* The generated fingerprints are decoded into potential molecules and converted into RDKit molecular structures for downstream tasks like conformer generation and scoring.
* The VAE enhances the pipeline by exploring a range of structurally similar molecules, allowing for a broader search in the molecular space, which is especially valuable in drug discovery.

**2. OpenVINO for Optimized Inference**

* The **OpenVINO model** is used to score each molecule by predicting binding affinity or lipophilicity based on the generated fingerprints.
* The OpenVINO runtime compiles and optimizes a model for efficient inference on Intel CPUs, reducing latency and improving throughput for large-scale molecule evaluations.
* This setup enables real-time scoring of generated molecules, allowing the pipeline to quickly rank and select promising candidates.

**3. Intel Extension for PyTorch (IPEX)**

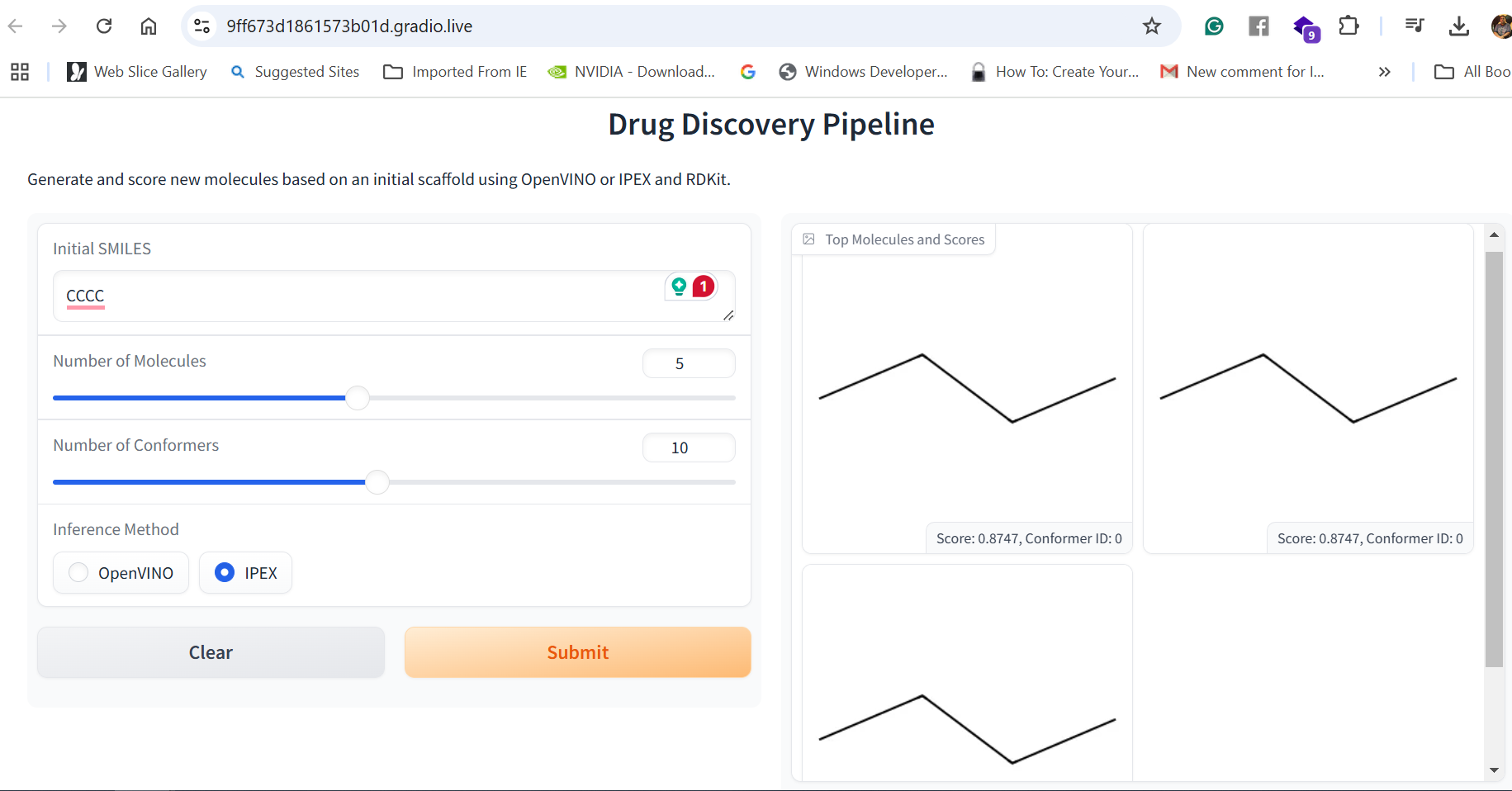
* **IPEX** optimizes the PyTorch model for binding affinity or lipophilicity prediction. This version of the model is used as an alternative to OpenVINO, providing an optimized execution path specifically for Intel hardware.
* Users can select either OpenVINO or IPEX for inference, allowing flexibility depending on the hardware environment and performance requirements.

**4. Gradio for Interactive Exploration**

* The pipeline is wrapped in a **Gradio interface**, enabling users to input an initial SMILES string, select the number of molecules and conformers to generate, and choose between OpenVINO or IPEX for inference.
* Gradio displays the top-ranked molecules with conformer scores, providing a user-friendly way to explore and visualize the generated molecules and their properties interactively.

**Summary**

The code combines GenAI for molecule generation, OpenVINO and IPEX for efficient inference, and Gradio for interactive exploration, creating a robust pipeline suited for drug discovery applications. This pipeline allows users to generate, score, and visualize new molecular structures in real-time, making it a powerful tool for computational chemistry and drug discovery



Gradio Inputs to check

 **Simple Aromatic Compound (Benzene)**

* SMILES: C1=CC=CC=C1
* Description: Benzene is a simple aromatic ring. This input allows you to see how functional groups can be added to a stable, small ring structure.

 **Small Alkane Chain (Butane)**

* SMILES: CCCC
* Description: Butane represents a basic alkane chain, allowing the generative model to build more complex hydrocarbons by adding functional groups.

 **Aromatic Amine (Aniline)**

* SMILES: C1=CC=C(C=C1)N
* Description: Aniline has both an aromatic ring and an amine group, which could lead to diverse modifications and generate interesting drug-like molecules.

 **Carboxylic Acid (Acetic Acid)**

* SMILES: CC(=O)O
* Description: A small carboxylic acid that could lead to the generation of molecules with acidic or basic properties.

 **Pyridine (Heteroaromatic Compound)**

* SMILES: C1=CC=NC=C1
* Description: Pyridine is a basic heteroaromatic compound commonly used in medicinal chemistry. This scaffold might result in biologically active compounds.



Step 4:

Predictions2.ipynb

The code in this notebook is focused on reaction prediction and employs **IPEX** and **OpenVINO** for model optimization and efficient inference. Here’s a breakdown of how each component is used in the notebook:

**1. Intel Extension for PyTorch (IPEX)**

* **IPEX Optimization for Training**: The model is optimized with IPEX during the training process. The ReactionPredictionModel is a neural network designed to predict the likelihood of a reaction outcome based on concatenated reactant and product molecular fingerprints.
* The model and optimizer are optimized together with IPEX, which utilizes Intel hardware optimizations to accelerate training on Intel CPUs.
* This helps in efficiently training the model on reaction prediction tasks, especially when handling large feature vectors.

**2. OpenVINO for Efficient Inference**

* After training, the model is reinitialized without IPEX optimizations and exported to ONNX format. This ONNX model is then loaded and compiled using OpenVINO, which optimizes it for fast inference on Intel CPUs.
* **Inference with OpenVINO**: The OpenVINO runtime is used to load and compile the ONNX model, which allows efficient inference by taking advantage of Intel-specific optimizations. This setup is particularly useful for deploying the model in production for real-time reaction prediction.

**3. Visualization of Results**

* The code includes visualization of the molecular structures (reactants and products) and their prediction likelihoods using RDKit and IPython display capabilities.
* Although **Gradio** is not directly used in this notebook, the visualization steps could be adapted to Gradio for an interactive user interface, allowing users to input reactant SMILES strings and view predicted reaction likelihoods alongside molecular images.

**Summary**

* **IPEX** is used to optimize the model during training for improved performance on Intel hardware.
* **OpenVINO** is leveraged for efficient, optimized inference by compiling the trained ONNX model for real-time reaction likelihood predictions.
* The setup creates a streamlined pipeline where IPEX-accelerated training is followed by OpenVINO-based inference, providing a fast, deployable solution for reaction prediction tasks in chemistry.

Step 5 :RAGMain.ipynb

**1. RAG (Retrieval-Augmented Generation) with Qdrant**

* **Vector Database Setup:**
  + **The code initializes a Qdrant client (client = QdrantClient(":memory:")) to store reaction vectors in an in-memory database, meaning this setup is temporary and useful for local testing.**
  + **The database collection, named "reactions", is configured to store 4096-dimensional vectors representing chemical structures. Cosine similarity is chosen for distance measurement, which is effective for similarity searches in this context.**
* **Fingerprint Vectorization:**
  + **Each molecule’s structure is converted into a fingerprint vector using RDKit’s GetMorganGenerator. This generator creates a 4096-dimensional fingerprint that encodes structural characteristics of the molecule.**
  + **These vectors, which capture unique features of the reactant molecules, are stored in Qdrant as numpy arrays. This approach ensures each reactant is represented as a vector in the database for efficient similarity searches.**
* **Database Population:**
  + **The function populate\_vector\_db\_in\_batches iterates through each reaction in the dataset, converts the SMILES notation of reactants into fingerprint vectors, and adds them to the Qdrant database.**
  + **Each reaction’s data, including its unique ID, reactant and product SMILES, and likelihood score, is stored as a “payload” along with its fingerprint vector. This setup allows easy retrieval of related data during searches.**
* **Similarity Search:**
  + **The function retrieve\_similar\_reactions takes a query vector and performs a search in Qdrant to find the closest matches based on cosine similarity.**
  + **This function returns a list of the top similar reactions, offering users a historical context that helps interpret the predicted reaction likelihood by providing reference reactions with similar reactant structures.**

**2. OpenVINO for Reaction Prediction**

* **Model Loading:**
  + **The OpenVINO runtime is set up with ov.Core(), and a pre-trained model (production\_prediction\_openvino1.xml) is loaded specifically for reaction prediction.**
  + **This model was optimized in OpenVINO format to take combined reactant and product fingerprints as input and output a likelihood score, representing the feasibility of the reaction.**
* **Input Preparation:**
  + **The function predict\_reaction\_with\_rag\_openvino converts both reactant and product SMILES strings into fingerprint vectors. These vectors are concatenated to form a single combined vector.**
  + **This combined vector, representing both reactant and product molecules, is reshaped to meet the model’s input format of [1, 8192] and converted to the float32 data type as required by OpenVINO.**
* **Inference Execution:**
  + **The model performs inference on the combined vector using OpenVINO, predicting a likelihood score for the reaction.**
  + **This score provides a numerical estimation of the reaction’s potential feasibility based on learned patterns from historical data, offering insights into the confidence level of the reaction’s success.**

**3. IPEX (Intel Extension for PyTorch)**

* **IPEX Absence:**
  + **There is no direct use of Intel Extension for PyTorch (IPEX) in this code since OpenVINO is used for inference and Qdrant handles vector retrieval.**
  + **IPEX could be beneficial if the model had PyTorch-based preprocessing or was trained in PyTorch, as it can optimize PyTorch execution on Intel hardware. However, here, the model has already been converted to OpenVINO format, making IPEX unnecessary for this specific pipeline.**

**4. Gradio Serving**

* **User Interface (UI) Design:**
  + **Gradio’s Blocks API is used to create an interactive user interface. This interface includes input fields for the user to enter SMILES strings of the reactant and product molecules.**
  + **The UI design is simple and user-friendly, providing a straightforward way for users to interact with the model and explore chemical reactions.**
* **Output Layout:**
  + **After entering SMILES strings, the user clicks the “Predict Reaction Likelihood” button, which triggers the predict\_reaction\_with\_rag\_openvino function.**
  + **Gradio then displays:**
    - **Predicted Reaction Likelihood: A numerical score showing the predicted feasibility of the reaction.**
    - **Molecular Images: Visual representations of the reactant and product molecules, generated by RDKit.**
    - **Similar Reactions: A list of similar reactions retrieved from Qdrant, providing historical reference and context for the user’s query.**

**Gradio input test:**

**Example 1: Ethanol to Acetaldehyde (Oxidation Reaction)**

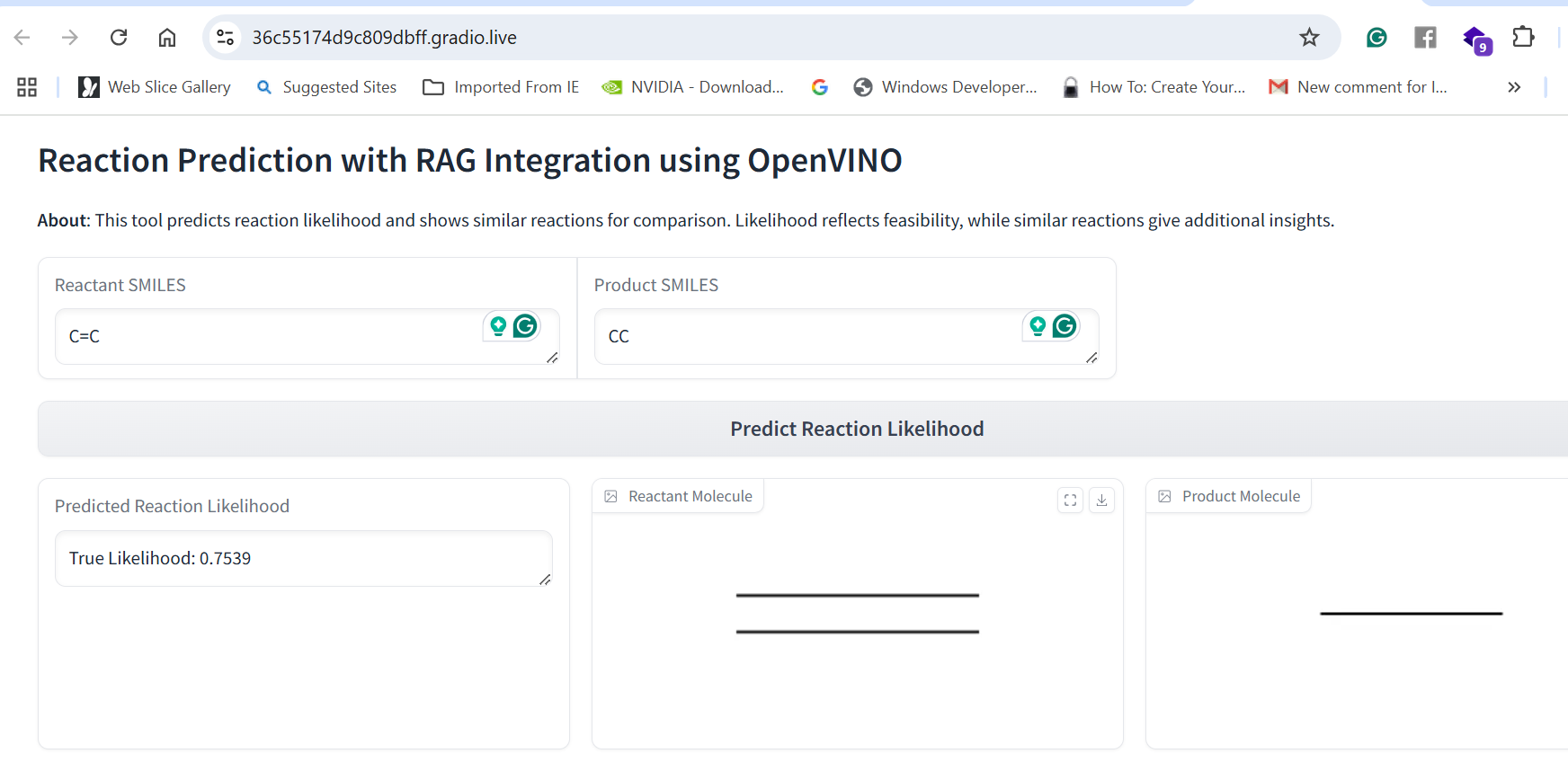
* **Reactant SMILES: CCO (ethanol)**
* **Product SMILES: CC=O (acetaldehyde)**

**Example 2: Benzene to Bromobenzene (Aromatic Substitution Reaction)**

* **Reactant SMILES: c1ccccc1 (benzene)**
* **Product SMILES: c1ccc(cc1)Br (bromobenzene)**

**Example 3: Ethene to Ethane (Hydrogenation Reaction)**

* **Reactant SMILES: C=C (ethene/ethylene)**
* **Product SMILES: CC (ethane)**



**The** entire workflow for the application is shown below.

**Step 1: Lipophilicity Prediction**

* **Objective**: Assess the lipophilicity (or hydrophobic/hydrophilic characteristics) of drug-like molecules, which influences absorption, distribution, and solubility, and is crucial for determining pharmacokinetic properties.
* **Procedure**:
  + **Data Preparation**: Use a dataset of molecules with known lipophilicity values, represented in SMILES format, to capture relevant chemical information.
  + **Modeling**: Deploy an OpenVINO-optimized regression model trained to predict lipophilicity. The model takes the molecular structure as input and outputs a predicted lipophilicity score.
  + **Deployment**: Containerize the model and deploy it on a scalable platform like OpenShift, enabling real-time lipophilicity predictions across large molecule libraries.
  + **Output**: The application provides predicted lipophilicity scores, allowing researchers to filter molecules based on solubility and absorption potential, which is key to initial drug screening.

**Step 2: Binding Affinity Prediction**

* **Objective**: Predict the binding affinity between a drug molecule and a target protein, which is a critical factor in determining a drug’s efficacy.
* **Procedure**:
  + **Data Preparation**: Compile a dataset of protein-ligand pairs with known binding affinities, with ligands represented in SMILES format and protein structures in 3D format (PDB or similar).
  + **Modeling**: Utilize an OpenVINO-optimized binding affinity model to predict interaction strength between molecules and target proteins. This model is designed to assess how well a drug molecule binds to its target, a critical measure of its potential efficacy.
  + **Deployment**: Deploy the binding affinity model on OpenShift for scalable and efficient processing, supporting high-throughput screening of compound libraries.
  + **Output**: Predicted binding affinity scores allow ranking of molecules by their likelihood of strong binding interactions with the target protein, enabling prioritization of promising candidates for further testing.

**Step 3: Drug Discovery Conformers Generation and Scoring**

* **Objective**: Generate multiple conformations (3D orientations) of potential drug molecules and score them to identify the most likely stable and effective conformers.
* **Procedure**:
  + **Scaffold Selection**: Define an initial molecular scaffold based on prior screening (such as from lipophilicity or binding affinity analysis).
  + **Conformer Generation**: Use RDKit to generate a range of 3D conformations for each molecule, allowing the exploration of structural variations that may impact binding and efficacy.
  + **Conformer Scoring**: Score each conformer for stability, binding affinity, and other relevant pharmacokinetic properties, using models optimized with OpenVINO to accelerate inference.
  + **Deployment**: Containerize this conformer generation and scoring workflow and deploy on OpenShift, allowing parallel processing of multiple candidates.
  + **Output**: Top-scoring conformers are identified, providing researchers with structural variants that maximize stability and binding potential, facilitating targeted design of effective drug candidates.

**Step 4: Reaction Prediction and SMILES Visualization**

* **Objective**: Predict chemical reactions and visualize resulting products to assess synthetic feasibility, helping optimize the drug synthesis pathway.
* **Procedure**:
  + **Reaction Prediction**: Using an OpenVINO-optimized reaction prediction model, input SMILES representations of reactants to predict possible reaction products. The model relies on patterns learned from known reaction pathways.
  + **Visualization**: RDKit generates visual representations of SMILES, converting them into molecular images that chemists can analyze. These visuals provide an accessible view of reaction outcomes.
  + **Deployment**: Deploy reaction prediction and visualization workflows on OpenShift for real-time feedback, allowing users to quickly assess synthetic feasibility and pathway optimization.
  + **Output**: Predicted products and their visualizations assist researchers in selecting viable reactions for synthesizing new drug compounds, providing insights into practical synthesis routes.

**Step 5: Reaction Prediction with Retrieval-Augmented Generation (RAG) Integration**

* **Objective**: Improve reaction prediction accuracy by using RAG to integrate retrieval-based contextual data with generative models, enhancing prediction quality with historical reaction data.
* **Procedure**:
  + **Data Retrieval**: Qdrant serves as a vector database storing reaction data as molecular fingerprints. During prediction, the application retrieves similar reactions from the database based on structural similarity, offering context from historical reactions.
  + **Reaction Prediction with RAG**: The OpenVINO model uses both the input reactant and retrieved examples to predict the reaction product. This hybrid approach improves prediction accuracy by providing the model with additional reference data.
  + **Deployment on OpenShift**: The RAG-enabled reaction prediction model is deployed on OpenShift to handle large-scale retrieval and generation requests efficiently, supporting high-throughput synthesis planning.
  + **Output**: The application outputs more accurate and contextually enriched reaction predictions, presenting researchers with synthesized pathways that are feasible and informed by historical data, further supporting data-driven decision-making.

### **How AMX is Utilized in the Code**

**Intel Advanced Matrix Extensions (AMX)** are hardware instructions designed to accelerate workloads involving large matrix operations. While AMX is primarily utilized internally by optimized libraries like Intel® Extension for PyTorch (IPEX), you can explicitly enable AMX by ensuring proper configuration and leveraging specific data types such as bfloat16.

Here’s an analysis of how AMX is used in the provided code:

### **1. Data Type Optimization**

* **bfloat16 Data Type**:
  + AMX instructions are optimized for bfloat16, which reduces memory bandwidth and accelerates matrix computations.
  + In the code, the following ensures that computations leverage bfloat16

model = model.to(torch.bfloat16)

features, labels = features.to(torch.bfloat16), labels.to(torch.bfloat16)

By converting the model parameters, input tensors, and gradients to bfloat16, matrix multiplications (e.g., Linear layers) can use AMX.

### **2. IPEX Optimization**

* **Intel® Extension for PyTorch (IPEX)**:
  + IPEX automatically optimizes the computation graph and operations for Intel hardware, including enabling AMX when available.
  + The following line explicitly requests bfloat16 optimizations for both the model and the optimizer:

model, optimizer = ipex.optimize(model, optimizer=optimizer, dtype=torch.bfloat16)

**How this uses AMX**:

* IPEX replaces default PyTorch operations with optimized kernels that use AMX when bfloat16 operations are detected in layers such as Linear, Conv, and matrix multiplications.

### **3. Fused Kernels for Optimizer**

* **Adam Optimizer with Fused Kernels**:
  + When AMX is enabled through IPEX, operations in the Adam optimizer (e.g., updates for momentum and variance) are fused into a single kernel that utilizes AMX for efficiency.
  + The error adam\_fused\_step\_kernel was resolved by ensuring that all parameters and optimizer states were in bfloat16, enabling the fused kernel to leverage AMX.

### **4. Matrix Operations in Layers**

* **Linear Layers (nn.Linear)**:
  + The fully connected layers (e.g., fc1, fc2, etc.) in the model involve matrix multiplications

x = torch.relu(self.fc1(x))

IPEX optimizes these multiplications to use AMX when both the weights and input tensors are in bfloat16.

These multiplications are optimized by IPEX to use AMX when both the weights and input tensors are in bfloat16.

Conclusion:-

AMX is explicitly utilized in the code via:

* Conversion of data and model parameters to bfloat16.
* Optimization of the computation graph using IPEX with dtype=torch.bfloat16.
* Automatic execution of fused kernels and matrix operations optimized for AMX by IPEX.

These steps enable the harnessing of AMX capabilities for significant performance improvements in matrix-heavy operations. Let me know if you want further clarification or help with profiling tools!