Exp No. 9	**Machine Learning Interoperability with ONNX**	**REG. NO: URK23CS1197**
Date:		

Objective:

The objective is to enable seamless interoperability between different machine learning frameworks and tools by using the ONNX (Open Neural Network Exchange) standard. This allows models to be easily shared, transferred, and deployed across various platforms without the need for retraining or significant modifications.

Tools/ Software Required:

- scikit-learn
- onnx
- onnxruntime
- skl2onnx

Job Role:

Machine Learning Engineer / Data Scientist

Skills Required:

- **Proficiency in Machine Learning Frameworks:** Familiarity with frameworks like TensorFlow, PyTorch, and Scikit-learn.
- **Understanding of ONNX:** Knowledge of the ONNX standard, including how to convert models to and from ONNX format.
- **Python Programming:** Strong skills in Python for implementing and manipulating machine learning models.
- **Model Conversion Tools:** Experience with tools like ONNX Runtime, ONNX.js, and other libraries for converting and running ONNX models.

Prerequisites:

- **Basic Understanding of Machine Learning:** Knowledge of machine learning concepts and algorithms.
- **Experience with Machine Learning Frameworks**: Prior experience with at least one machine learning framework.
- **Familiarity with Python:** Comfortable working with Python for data manipulation and analysis.
- **Basic Knowledge of ONNX:** Understanding of the ONNX format and its benefits for interoperability.

Description:

Machine learning interoperability refers to the ability of different machine learning systems, models, or components to work together seamlessly

Model Interoperability: Ensuring that machine learning models trained in one framework or platform can be used in another. Common model formats like ONNX (Open Neural Network Exchange) are designed to facilitate model interoperability.

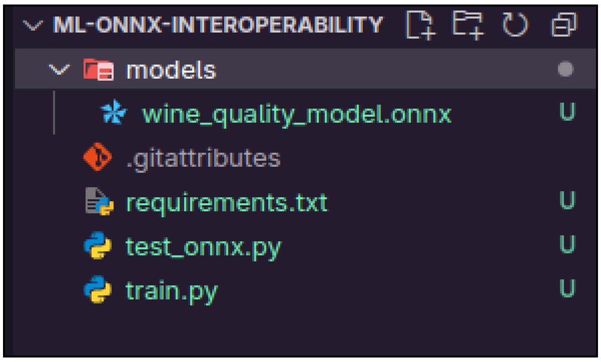
ONNX

ONNX, which stands for Open Neural Network Exchange, is an open-source format and ecosystem designed to enable interoperability among various deep learning frameworks and machine learning tools.

Program:

Project Structure:

```
ML-ONNX-Interoperability/
├── models/
│   └── wine_quality_model.onnx
├── train.py
└── test_onnx.py
```



```
ML-ONNX-INTEROPERABILITY
  models
    wine_quality_model.onnx   U
  .gitattributes
  requirements.txt            U
  test_onnx.py                U
  train.py                    U
```

requirements.txt

```
scikit-learn
onnx
onnxruntime
skl2onnx
```

train.py

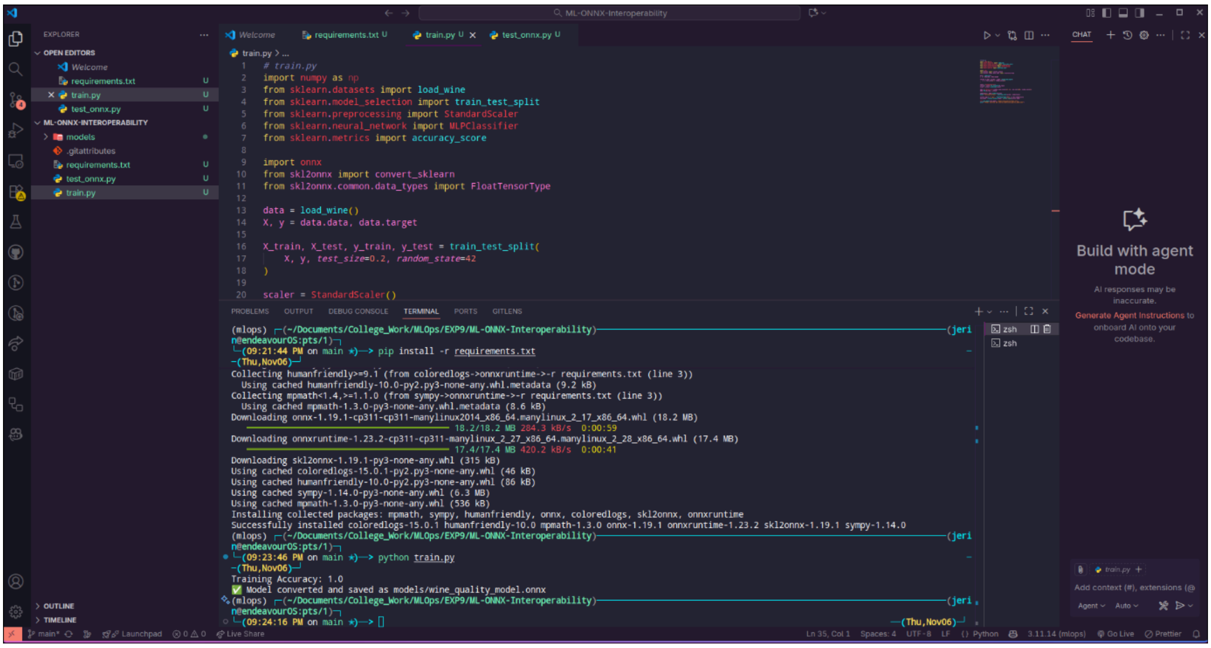
```
# train.py
import numpy as np
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

```
import onnx
from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import FloatTensorType

data = load_wine()
X, y = data.data, data.target

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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=1000, random_state=42)
model.fit(X_train, y_train)

predictions = model.predict(X_test)
print("Training Accuracy:", accuracy_score(y_test, predictions))

initial_type = [('input', FloatTensorType([None, X_train.shape[1]]))]
onnx_model = convert_sklearn(model, initial_types=initial_type)

onnx.save_model(onnx_model, "models/wine_quality_model.onnx")
print("✅ Model converted and saved as models/wine_quality_model.onnx")
```

test_onnx.py

```
# test_onnx.py
import numpy as np
import onnx
import onnxruntime as ort

providers = ['CPUExecutionProvider']
session = ort.InferenceSession("models/wine_quality_model.onnx", providers=providers)

input_data = np.array([
    [13.2, 3.3, 2.3, 21.0, 105.0, 2.5, 2.3, 0.27, 1.35, 4.0, 1.0, 3.2, 750]
], dtype=np.float32)

output = session.run(None, {"input": input_data})
prediction = int(np.argmax(output[1]))
print("Prediction (Numeric Class):", prediction)

class_labels = ["Class 0", "Class 1", "Class 2"]
print("Predicted Wine Category:", class_labels[prediction])
```

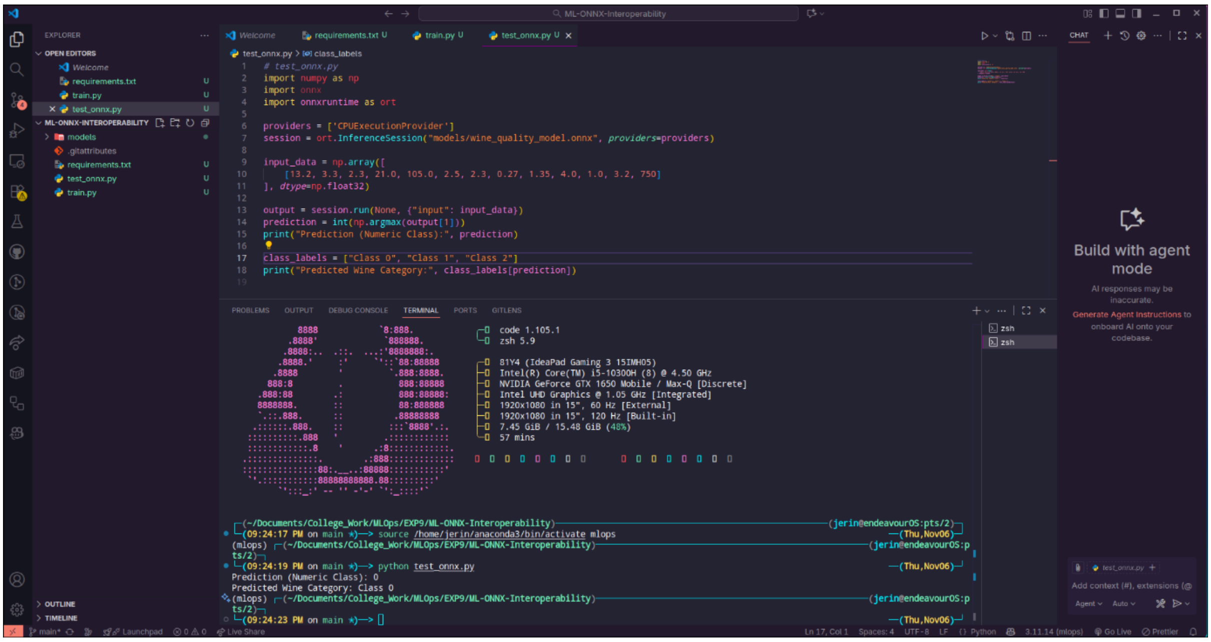
Expected Output :



train.py > ...

```python
# train.py
import numpy as np
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

import onnx
from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import FloatTensorType

data = load_wine()
X, y = data.data, data.target

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

scaler = StandardScaler()
```

```
(mlops) ┌─(~/Documents/College_Work/MLOps/EXP9/ML-ONNX-Interoperability)──────────────────────(jeri
n@endeavourOS:pts/1)─┐
└─(09:21:44 PM on main ★)──> pip install -r requirements.txt
─(Thu,Nov06)─┘
Collecting humanfriendly>=9.1 (from coloredlogs->onnxruntime->-r requirements.txt (line 3))
  Using cached humanfriendly-10.0-py2.py3-none-any.whl.metadata (9.2 kB)
Collecting mpmath<1.4,>=1.1.0 (from sympy->onnxruntime->-r requirements.txt (line 3))
  Using cached mpmath-1.3.0-py3-none-any.whl.metadata (8.6 kB)
Downloading onnx-1.19.1-cp311-cp311-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (18.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 284.3 kB/s  0:00:59
Downloading onnxruntime-1.23.2-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (17.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.4/17.4 MB 420.2 kB/s  0:00:41
Downloading skl2onnx-1.19.1-py3-none-any.whl (315 kB)
Using cached coloredlogs-15.0.1-py2.py3-none-any.whl (46 kB)
Using cached humanfriendly-10.0-py2.py3-none-any.whl (86 kB)
Using cached sympy-1.14.0-py3-none-any.whl (6.3 MB)
Using cached mpmath-1.3.0-py3-none-any.whl (536 kB)
Installing collected packages: mpmath, sympy, humanfriendly, onnx, coloredlogs, skl2onnx, onnxruntime
Successfully installed coloredlogs-15.0.1 humanfriendly-10.0 mpmath-1.3.0 onnx-1.19.1 onnxruntime-1.23.2 skl2onnx-1.19.1 sympy-1.14.0
(mlops) ┌─(~/Documents/College_Work/MLOps/EXP9/ML-ONNX-Interoperability)──────────────────────(jeri
n@endeavourOS:pts/1)─┐
└─(09:23:46 PM on main ★)──> python train.py
─(Thu,Nov06)─┘
Training Accuracy: 1.0
✅ Model converted and saved as models/wine_quality_model.onnx
(mlops) ┌─(~/Documents/College_Work/MLOps/EXP9/ML-ONNX-Interoperability)──────────────────────(jeri
n@endeavourOS:pts/1)─┐
└─(09:24:16 PM on main ★)──> ▯                                                   ──(Thu,Nov06)─┘
```



test_onnx.py > class_labels

```python
# test_onnx.py
import numpy as np
import onnx
import onnxruntime as ort

providers = ['CPUExecutionProvider']
session = ort.InferenceSession("models/wine_quality_model.onnx", providers=providers)

input_data = np.array([
    [13.2, 3.3, 2.3, 21.0, 105.0, 2.5, 2.3, 0.27, 1.35, 4.0, 1.0, 3.2, 750]
], dtype=np.float32)

output = session.run(None, {"input": input_data})
prediction = int(np.argmax(output[1]))
print("Prediction (Numeric Class):", prediction)

class_labels = ["Class 0", "Class 1", "Class 2"]
print("Predicted Wine Category:", class_labels[prediction])
```

```
              8888           `8:888.           ┌─▯ code 1.105.1
            .8888'            `888888.         └─▯ zsh 5.9
           .8888:..   .::.  ...:'8888888:.
          .8888.'     :'     `'::`88:88888     ┌─▯ 81Y4 (IdeaPad Gaming 3 15IMH05)
         .8888        '        `.888:8888.     ├─▯ Intel(R) Core(TM) i5-10300H (8) @ 4.50 GHz
        888:8         .          888:88888     ├─▯ NVIDIA GeForce GTX 1650 Mobile / Max-Q [Discrete]
       .888:88       .:          888:88888:    ├─▯ Intel UHD Graphics @ 1.05 GHz [Integrated]
       8888888.      ::          88:888888     ├─▯ 1920x1080 in 15", 60 Hz [External]
        `.::.888.    ::         .88888888      ├─▯ 1920x1080 in 15", 120 Hz [Built-in]
      .::::::.888.   ::        :::`8888'.:.    ├─▯ 7.45 GiB / 15.48 GiB (48%)
     ::::::::::.888   '       .::::::::::::    └─▯ 57 mins
     ::::::::::::.8    '     .:8:::::::::::::.
    .::::::::::::::.        .:888:::::::::::::   ▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯     ▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯
    :::::::::::::::88:.__..:88888:::::::::::'
     `'.:::::::::::88888888888.88:::::::::'
           `':::_:' -- '' -'-' `':_::::'`

┌─(~/Documents/College_Work/MLOps/EXP9/ML-ONNX-Interoperability)──────────────(jerin@endeavourOS:pts/2)─┐
└─(09:24:17 PM on main ★)──> source /home/jerin/anaconda3/bin/activate mlops         ──(Thu,Nov06)─┘
(mlops) ┌─(~/Documents/College_Work/MLOps/EXP9/ML-ONNX-Interoperability)────────────────(jerin@endeavourOS:p
ts/2)─┐
└─(09:24:19 PM on main ★)──> python test_onnx.py                                     ──(Thu,Nov06)─┘
Prediction (Numeric Class): 0
Predicted Wine Category: Class 0
(mlops) ┌─(~/Documents/College_Work/MLOps/EXP9/ML-ONNX-Interoperability)────────────────(jerin@endeavourOS:p
ts/2)─┐
└─(09:24:23 PM on main ★)──> ▯                                                       ──(Thu,Nov06)─┘
```

Result :

The trained Wine Quality classification model was successfully converted into the ONNX format and executed using ONNX Runtime. This demonstrates seamless interoperability between machine learning frameworks, allowing the model to run independently of its original training environment.