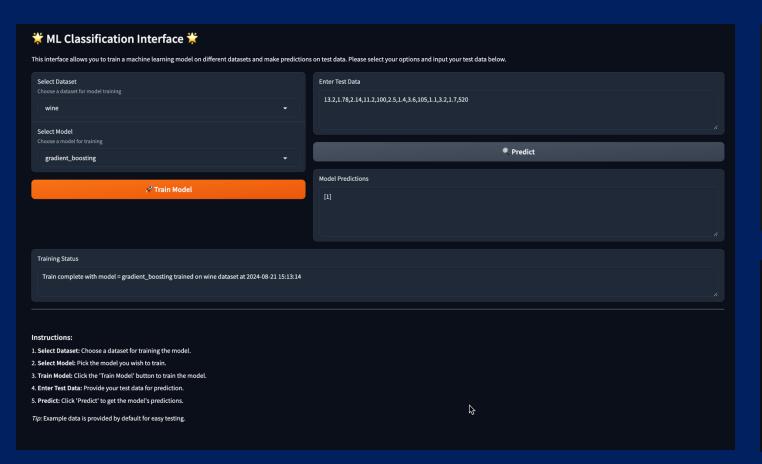
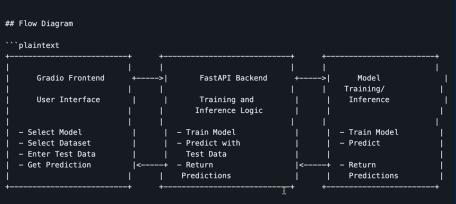
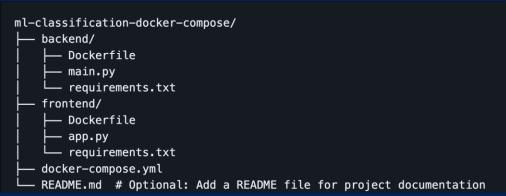


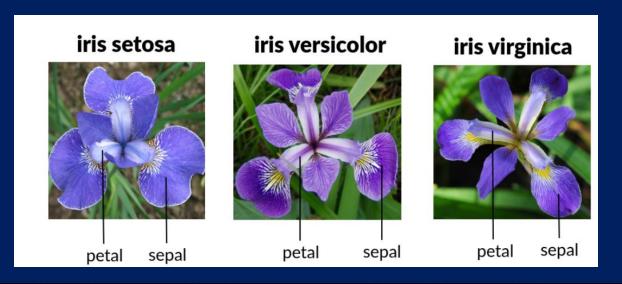
SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS







```
version: '3.8'
services:
  backend:
    build: ./backend
    container_name: backend
    ports:
     - "8084:8000"
    volumes:
     - ./backend:/app
  frontend:
    build: ./frontend
    container_name: frontend
    ports:
      - "8085:7860"
    depends_on:
     backend
    volumes:
     - ./frontend:/app
```



Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa

Name: Red Wine Quality Data Set

• Source: UCI Machine Learning Repository

• Input variables:
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide

7 - total sulfur dioxide

8 - density 9 - pH

10 - sulphates 11 - alcohol

• Output variable: quality (score between 0 and 10)

• Data Set Characteristics: Multivariate

• Number of Observations: 1599

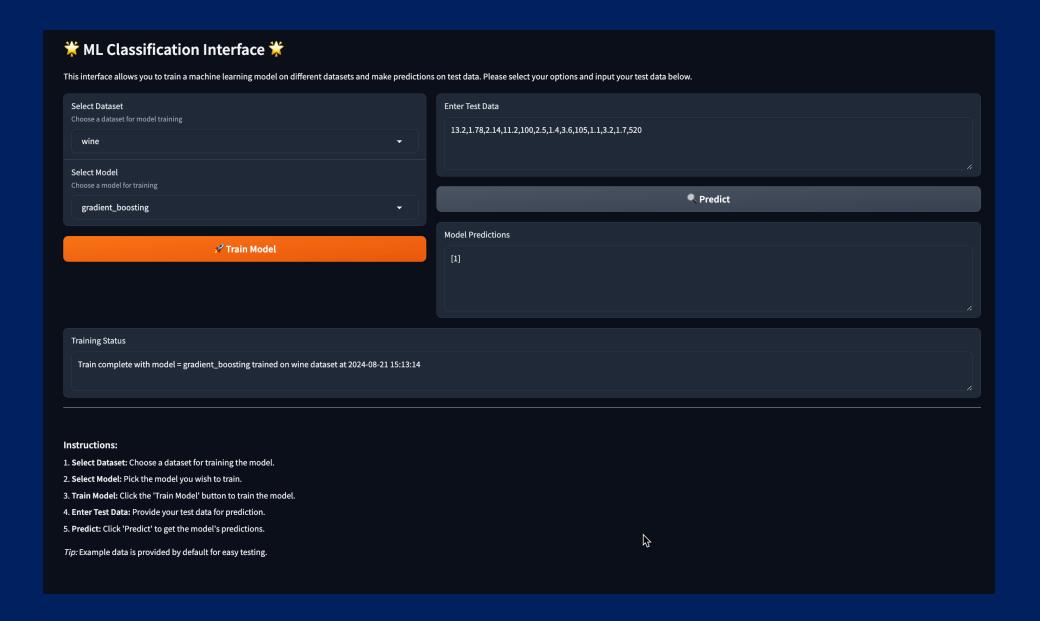
• Number of Attributes/Variables: 12

Missing Values: N/A



Source: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	рН	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5



Back end

```
@app.post("/train/")
async def train_model(request: TrainRequest):
    if request.dataset_name not in datasets:
        raise HTTPException(status_code=404, detail="Dataset not found")
   if request.model_name not in ["random_forest", "gradient_boosting"]:
        raise HTTPException(status_code=400, detail="Invalid model name")
   # Load dataset
   data = datasets[request.dataset_name]
   X = data.data
   y = data.target
    print(X.shape, y.shape)
    # Split dataset
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    # Select model
    if request.model_name == "random_forest":
        model = RandomForestClassifier()
   elif request.model_name == "gradient_boosting":
        model = GradientBoostingClassifier()
   # Train model
   model.fit(X_train, y_train)
   # Save the model
    models[request.model_name] = model
   # Format current time as a string
    current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
    return {"message": f"Train complete with model = {request.model_name} trained on {request.dataset_name} dataset at {current_time}"}
@app.post("/predict/")
async def predict(request: PredictRequest):
    if request.model_name not in models:
        raise HTTPException(status_code=404, detail="Model not found")
    if request.model_name in models:
        model = models[request.model_name]
        # Predict using the provided test data
        predictions = model.predict(request.test_data)
        return {"predictions": predictions.tolist()}
    else:
        raise HTTPException(status_code=400, detail="Model not found")
```

