Cancer Detection using Machine Learning (Python)

Repository Structure

```
cancer-detection-ml/
■■ LICENSE
■■ README.md
■■ requirements.txt
■■ .gitignore
■■ src/
■   ■■ train.py
■   ■■ predict.py
■   ■■ utils.py
■■ models/
■   ■■ (model artifacts saved here)
■■ notebooks/
   ■■ analysis.ipynb
```

src/utils.py

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

def load_data(as_frame=True, test_size=0.2, random_state=42):
    from sklearn.datasets import load_breast_cancer
    ds = load_breast_cancer(as_frame=as_frame)
    X = ds.data
    y = ds.target
    return train_test_split(X, y, test_size=test_size, random_state=random_state)
```

src/train.py

```
import argparse
import joblib
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
from src.utils import load_data

def main(output_path: str):
    X_train, X_test, y_train, y_test = load_data()

    pipe = Pipeline([
        ("scaler", StandardScaler()),
        ("clf", RandomForestClassifier(random_state=42))
    ])

    param_grid = {
        'clf__n_estimators': [50, 100],
        'clf__max_depth': [None, 8, 16],
        'clf__min_samples_split': [2, 5]
    }

    gs = GridSearchCV(pipe, param_grid, cv=5, scoring='f1', n_jobs=-1)
    gs.fit(X_train, y_train)

    print("Best params:", gs.best_params_)
    y_pred = gs.predict(X_test)
    print("Classification report:", classification_report(y_test, y_pred))
    print("Confusion matrix:", confusion_matrix(y_test, y_pred))
```

```
    joblib.dump(gs.best_estimator_, output_path)
    print(f"Saved best model to {output_path}")

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--output', type=str, default='models/model.joblib')
    args = parser.parse_args()
    main(args.output)
```

Expected Output (Training)

```
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2, 'clf__n_estimators': 100}

Classification report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        43
           1       0.97      0.99      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Confusion matrix:
[[41  2]
 [ 1 70]]

Saved best model to models/model.joblib
```

src/predict.py

```
import argparse
import joblib
import numpy as np
from src.utils import load_data

def main(model_path: str, n_samples: int = 5):
    model = joblib.load(model_path)
    X_train, X_test, y_train, y_test = load_data()
    X_sample = X_test[:n_samples]
    y_true = y_test[:n_samples]

    y_pred = model.predict(X_sample)
    for i, (pred, true) in enumerate(zip(y_pred, y_true)):
        print(f"Sample {i}: predicted={pred} (0=malignant,1=benign), true={true}")

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--model', type=str, default='models/model.joblib')
    args = parser.parse_args()
    main(args.model)
```

Expected Output (Prediction)

```
Sample 0: predicted=1 (0=malignant,1=benign), true=1
Sample 1: predicted=0 (0=malignant,1=benign), true=0
Sample 2: predicted=1 (0=malignant,1=benign), true=1
Sample 3: predicted=1 (0=malignant,1=benign), true=1
Sample 4: predicted=0 (0=malignant,1=benign), true=0
```