ML Assignment 2 - Submission Report (Template)

1) GitHub Repository Link: _____
2) Live Streamlit App Link: _____
3) BITS Virtual Lab Screenshot: (paste in final PDF)

--- README CONTENT (copy-paste) ---

ML Assignment 2 – Multiple Classifiers + Streamlit App

a) Problem statement
Build and compare multiple machine-learning classification models on **one** public dataset, compute
evaluation metrics, and demonstrate the models in an interactive **Streamlit** web application. The ap
supports uploading test CSV data, selecting a model, viewing evaluation metrics, and viewing a confu
matrix / classification report.

b) Dataset description
Dataset: *Breast Cancer Wisconsin (Diagnostic)* (UCI Machine Learning Repository).

- Task: Binary classification (malignant vs benign).
- Size: **569 instances** and **30 numeric features** (meets the assignment minimums).
- Feature origin: Computed from digitized images of fine needle aspirate (FNA) of a breast mass; featu
describe characteristics of the cell nuclei.

References:
- UCI dataset page: https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
- scikit-learn loader (notes it is a copy of the UCI dataset): https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

Target encoding used in this repo: `target = 1` for **malignant**, `target = 0` for **benign**.

c) Models used + evaluation metrics
Models implemented (all on the same dataset):
1. Logistic Regression
2. Decision Tree Classifier
3. K-Nearest Neighbor (kNN)
4. Naive Bayes (Gaussian)
5. Random Forest (Ensemble)
6. XGBoost (Ensemble)

Metrics computed (as required): Accuracy, AUC, Precision, Recall, F1 score, Matthews Correlation
Coefficient (MCC).

Comparison table (holdout test set)
The table below is auto-generated by `src/train.py` and saved to `model/model_comparison_metrics.c

ML Model Name	Accuracy	AUC	Precision	Recall	F1	MCC
Logistic Regression	0.9649	0.9960	0.9750	0.9286	0.9512	0.9245
Decision Tree	0.9298	0.9246	0.9048	0.9048	0.9048	0.8492
kNN	0.9561	0.9825	0.9744	0.9048	0.9383	0.9058
Naive Bayes	0.9386	0.9934	1.0000	0.8333	0.9091	0.8715

| Random Forest (Ensemble) | 0.9737 | 0.9944 | 1.0000 | 0.9286 | 0.9630 | 0.9442 |
| XGBoost (Ensemble) | 0.9649 | 0.9924 | 1.0000 | 0.9048 | 0.9500 | 0.9258 |

Observations on model performance

ML Model Name	Observation about model performance
Logistic Regression	Performs best overall on this dataset (high Accuracy/AUC and balanced Precision-Recall).
Decision Tree	A single tree can fit non-linear patterns but may overfit; performance is usually below ensembles.
kNN	Sensitive to feature scaling and local neighborhood structure; works well when classes are well-separated.
Naive Bayes	Fast baseline; assumes feature independence which may limit performance when features are correlated.
Random Forest (Ensemble)	Performs best overall on this dataset (high Accuracy/AUC and balanced Precision-Recall).
XGBoost (Ensemble)	Boosted trees often achieve top performance by sequentially correcting errors from previous trees.
Logistic Regression	0.9649	0.9960	0.9750	0.9286	0.9512	0.9245
Decision Tree	0.9298	0.9246	0.9048	0.9048	0.9048	0.8492
kNN	0.9561	0.9825	0.9744	0.9048	0.9383	0.9058
Naive Bayes	0.9386	0.9934	1.0000	0.8333	0.9091	0.8715
Random Forest (Ensemble)	0.9737	0.9944	1.0000	0.9286	0.9630	0.9442
XGBoost (Ensemble)	0.9649	0.9924	1.0000	0.9048	0.9500	0.9258

Observations on model performance

ML Model Name	Observation about model performance
Logistic Regression	Performs best overall on this dataset (high Accuracy/AUC and balanced Precision-Recall).
Decision Tree	A single tree can fit non-linear patterns but may overfit; performance is usually below ensembles.
kNN	Sensitive to feature scaling and local neighborhood structure; works well when classes are well-separated.
Naive Bayes	Fast baseline; assumes feature independence which may limit performance when features are correlated.
Random Forest (Ensemble)	Performs best overall on this dataset (high Accuracy/AUC and balanced Precision-Recall).
XGBoost (Ensemble)	Boosted trees often achieve top performance by sequentially correcting errors from previous trees.

Project structure

```
project-folder/
  app.py
  requirements.txt
  README.md
```

```
  model/
    *.joblib
    model_comparison_metrics.csv
    artifacts.json
    holdout_test_set.csv
  data/
    breast_cancer_wdbc.csv
  src/
    data.py
    models.py
    metrics.py
    train.py
```

How to run locally
```bash
pip install -r requirements.txt
streamlit run app.py
```

How to deploy on Streamlit Community Cloud
1. Push this folder to a GitHub repository.
2. Go to https://streamlit.io/cloud and create a new app.
3. Select the repo, branch (main), and `app.py`.
4. Deploy.

Notes for assignment submission
- Add the GitHub repo link and Streamlit app link to your final submission PDF.
- Run once on BITS Virtual Lab and include a screenshot as required.