REPORT

Question 1: Gaussian Naïve Bayes Classifier on the Adult Income Dataset

---

1. Objective

Build and evaluate a Gaussian Naïve Bayes model to predict whether an individual earns over $50 K/yr, using:

- One-hot encoding of categorical features
- Standard accuracy, precision, recall, F1-score, ROC-AUC on a hold-out test set
- 10-fold cross-validation on the training set
- Comparison to the null (majority-class) baseline
- Decision-threshold tuning to improve recall on the ">50K" class

---

2. Data Preparation

- **Files:** adult.data (train), adult.test (test)
- **Columns:** 14 predictors + income (<=50K / >50K)
- **Cleaning:**
  - Strip trailing "." from adult.test labels
  - Dropped rows with any "?" values
- **Encoding:**
  - One-hot encode all 8 categorical cols
  - Leave continuous cols (age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week) as is
- **Targets:** Binary y = 1 if income=='>50K', else 0

---

3. Model Training

- **Algorithm:** Gaussian NB with default priors
- **Train/Test split:** Provided split (no further hold-out)
- **Cross-validation:** 10-fold on the training set

4. Result

```
Default (0.5) → Acc=0.789 Prec=0.648 Rec=0.306 F1=0.416
Tuned (0.3) → Acc=0.788 Prec=0.640 Rec=0.308 F1=0.416
ROC-AUC: 0.8256077631328511
CV acc: mean 0.789 range 0.78 - 0.798
Null acc: 0.754

Classification Report (threshold=0.30):
              precision    recall  f1-score   support

       <=50K       0.81      0.94      0.87     11360
        >50K       0.64      0.31      0.42      3700

    accuracy                           0.79     15060
   macro avg       0.72      0.63      0.64     15060
weighted avg       0.77      0.79      0.76     15060
```

5. Discussion

- **Lift over baseline:** 78.9 % vs. 75.4 % null accuracy → meaningful improvement.
- **ROC-AUC of 0.826** shows the model can distinguish classes well, even if the default threshold is conservative.
- **Threshold tuning** (0.50 → 0.30) traded a slight drop in precision (0.648 → 0.640) for a negligible gain in recall (0.306 → 0.308), keeping F1 stable.
- **CV stability:** 10-fold CV accuracies vary only ~0.018, indicating low variance and good generalization.

---

6. Conclusions & Next Steps

1. **Threshold adjustment** can be used to prioritize recall (sensitivity) at the expense of precision when detecting the “>50K” group.
2. **One-hot encoding** was critical—label encoding on categoricals would violate GaussianNB’s continuous-feature assumption.
3. **Future improvements:**
    - We can use **CategoricalNB** for purely discrete data.
    - Incorporate **feature selection** or **feature engineering** (e.g. interaction terms).
    - We can try other classifiers (e.g. Logistic Regression with class weights, ensemble methods) for better recall on the minority class.

Question 2: Decision Tree Classification on Car Evaluation Dataset

---

1. Exploratory Data Analysis (EDA)

- **Dataset size:** 1 728 instances, 7 attributes
- **Attributes (all categorical):**
  - **buying:** vhigh, high, med, low
  - **maint:** vhigh, high, med, low
  - **doors:** 2, 3, 4, 5more
  - **persons:** 2, 4, more
  - **lug_boot:** small, med, big
  - **safety:** low, med, high
  - **class (target):** unacc, acc, good, vgood
- **Class distribution:**
  - unacc: 1 210 (70.1 %)
  - acc: 384 (22.2 %)
  - good: 69 (4.0 %)
  - vgood: 65 (3.8 %)

Observation: The dataset is moderately imbalanced—"unacc" dominates, while "good" and "vgood" are rare.

---

2. Data Preprocessing

1. **Feature/Target split**
   - **Features:** all columns except **class**
   - **Target:** the **class** column
2. **Encoding**
   - Applied **one-hot encoding** to all six categorical inputs.
     This yields binary indicator columns (e.g. buying_low, doors_2, safety_high, etc.).
   - Left **class** as a string label for classification.
3. **Train/Test split**
   - 80 % training, 20 % testing
   - **Stratified** on the target to preserve class proportions

---

3. Model Training

- **Algorithms:**
  1. **Decision Tree (Gini index)**
  2. **Decision Tree (Entropy / Information Gain)**
- **Hyperparameters:** all default (no pruning, full-depth trees)
- **Random state:** 42

---

4. Evaluation Metrics

Criterion	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
Gini	0.974	0.955	0.941	0.945
Entropy	0.990	0.970	0.972	0.972

- **Confusion matrix (Entropy)** on 346 test samples:

```
     Predicted →
     acc    good  unacc  vgood
vrai
acc    71     2     4      0
good    0    14     0      0
unacc   1     0   241      0
vgood   2     0     0     11
```

- **Classification report** confirms very high precision & recall across all classes, with "Entropy" slightly outperforming "Gini."

---

5. Comparison & Discussion

- Both trees achieve **excellent** performance, reflecting the "clean" categorical nature of the data.
- The **Entropy-based** tree edges out **Gini** by ~1.6 points of accuracy and ~0.03 of macro-F1.
- **Why Entropy wins:** Information gain often yields purer child nodes when splitting on categorical features, at the cost of marginally more computation.
- **Why Gini still good:** Gini is faster and yields nearly identical results in most practical scenarios.

---

6. Conclusion & Recommendations

- If **maximum predictive accuracy** is paramount—and training time is not a concern— we can use **criterion='entropy'**.
- If we prefer **speed** with negligible loss in accuracy, **criterion='gini'** is acceptable.
- For improved **interpretability**, we can consider pruning via parameters like max_depth or min_samples_leaf; you can often maintain > 95 % accuracy with a much shallower tree.
- We can explore ensemble methods (Random Forests or Gradient Boosting) or test one-vs-rest strategies to further boost recall on minority classes ("good" and "vgood").

Question 3: Decision Tree Classification on Diabetes Dataset

1. Objective

Build and evaluate a Decision Tree model to predict whether a patient has diabetes (Outcome = 1) or not (Outcome = 0), using a 90/10 train–test split and hyperparameter tuning.

2. Data & Experimental Setup

- **Dataset:** Diabetes.csv (Pima Indians Diabetes)
- **Features:** 8 clinical measurements (e.g. glucose, BMI, age, etc.)
- **Target:** Outcome (0 = non-diabetic, 1 = diabetic)
- **Split:** 90 % training (691 samples), 10 % testing (77 samples), stratified on Outcome

3. Baseline Model

- **Classifier:** Decision Tree Classifier(random_state=42) with default parameters
- **Test-set Performance:**
  - **Accuracy:** 0.740
  - **Precision (class 1):** 0.640
  - **Recall (class 1):** 0.593
  - **F1-Score (class 1):** 0.615
- **Support on Test Set:** 50 negatives (0), 27 positives (1)

The baseline tree captures just under 60 % of true diabetics (recall) and achieves 74 % overall accuracy.

4. Hyperparameter Tuning

- **Grid Search (5-fold CV)** over:
  - max_depth: [None, 3, 5, 7, 9]
  - min_samples_split: [2, 5, 10]
  - min_samples_leaf: [1, 2, 4]
- **Best Parameters Found:**

```
bash
CopyEdit
{'max_depth': 5,
 'min_samples_split': 2,
 'min_samples_leaf': 4}
```

5. Optimized Model

- **Classifier:** DecisionTreeClassifier(max_depth=5, min_samples_split=2, min_samples_leaf=4, random_state=42)
- **Test-set Performance:**

- **Accuracy:** 0.779
- **Precision (class 1):** 0.708
- **Recall (class 1):** 0.630
- **F1-Score (class 1):** 0.667

After tuning, overall accuracy rose to 77.9 %, with a 7.8-point lift in precision and a 3.7-point lift in recall on the diabetic class.

---

6. Discussion & Conclusions

- **Generalization improved**: Constraining tree depth (max_depth=5) and requiring at least four samples per leaf reduced overfitting and raised test accuracy from 74.0 % to 77.9 %.
- **Better minority-class detection**: F1-score for diabetics increased from 0.615 to 0.667, reflecting more balanced precision/recall.
- **Next steps**:
  - Consider **pruning** further or adding **ensemble methods** (Random Forest, Gradient Boosting) for even higher robustness.
  - Use **cost-sensitive learning** or **class weights** if missing diabetics (false negatives) carries higher real-world risk.