Initial Model Training Code, Model Validation and Evaluation Report

Project Name : Covid - 19 Infant Growth Analysis and Prediction

Data Split

The dataset was split into **training (75%)** and **testing (25%)** sets

Features

- Numerical features: age_months, height_cm, weight_kg, speech_score, milestone_score.
- Categorical feature: period (encoded using LabelEncoder)
- Target labels: Infant developmental outcome (encoded using LabelEncoder)

Three machine learning models were trained:

- TabPFNClassifier
- XGBClassifier

Initial Model Training Code :

```python
from tabpfn import TabPFNClassifier
```

```python
tabpfn = TabPFNClassifier()
tabpfn.fit(x_train, y_train)
```

```
TabPFNClassifier
TabPFNClassifier()
```

```python
y_pred_tabpfn = tabpfn.predict(x_test)
```

```python
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred_tabpfn)
accuracy
```

```
1.0
```

```python
from sklearn.metrics import confusion_matrix, classification_report

cm_tabpfn = confusion_matrix(y_test, y_pred_tabpfn)
print(cm_tabpfn)

cs_report_tabpfn = classification_report(y_test, y_pred_tabpfn)
print(cs_report_tabpfn)
```

```
[[49  0  0]
 [ 0 53  0]
 [ 0  0 48]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        49
           1       1.00      1.00      1.00        53
           2       1.00      1.00      1.00        48

    accuracy                           1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150
```

```python
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

# Initialize XGBoost model
xgb = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    random_state=42
)

# Train the model
# Use the encoded training labels (y_train_encoded) from the previous cell
xgb.fit(x_train, y_train)

# Make predictions
y_pred_xgb = xgb.predict(x_test)

# Evaluate the model
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print(f"Accuracy: {accuracy_xgb}")
```

```
Accuracy: 0.9866666666666667
```

```python
from sklearn.metrics import confusion_matrix, classification_report

cm_xgb = confusion_matrix(y_test, y_pred_xgb)
print(cm_xgb)

cs_report_xgb = classification_report(y_test, y_pred_xgb)
print(cs_report_xgb)
```

```
[[49  0  0]
 [ 2 51  0]
 [ 0  0 48]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        49
           1       1.00      0.96      0.98        53
           2       1.00      1.00      1.00        48

    accuracy                           0.99       150
   macro avg       0.99      0.99      0.99       150
weighted avg       0.99      0.99      0.99       150
```

Model Validation and Evaluation Report :

Model	Summary	Training and validation performance metrics

Model - 1 : TabPFNClassifier	`from tabpfn import TabPFNClassifier`
	`tabpfn = TabPFNClassifier()`
	`tabpfn.fit(x_train, y_train)`
	`TabPFNClassifier`
	`TabPFNClassifier()`
	`y_pred_tabpfn = tabpfn.predict(x_test)`
		`from sklearn.metrics import accuracy_score`
		`accuracy = accuracy_score(y_test, y_pred_tabpfn)`
		`accuracy`
		`1.0`
		`from sklearn.metrics import confusion_matrix, classification_report`
		`cm_tabpfn = confusion_matrix(y_test, y_pred_tabpfn)`
		`print(cm_tabpfn)`
		`cs_report_tabpfn = classification_report(y_test, y_pred_tabpfn)`
		`print(cs_report_tabpfn)`
		`[[49 0 0]`
		`[ 0 53 0]`
		`[ 0 0 48]]`
		`precision recall f1-score support`
		`0 1.00 1.00 1.00 49`
		`1 1.00 1.00 1.00 53`
		`2 1.00 1.00 1.00 48`
		`accuracy 1.00 150`
		`macro avg 1.00 1.00 1.00 150`
		`weighted avg 1.00 1.00 1.00 150`

Model - 2 : XGBClassifier

```python
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

# Initialize XGBoost model
xgb = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    random_state=42
)

# Train the model
# Use the encoded training labels (y_train_encoded) from the previous cell
xgb.fit(x_train, y_train)

# Make predictions
y_pred_xgb = xgb.predict(x_test)

# Evaluate the model
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print(f"Accuracy: {accuracy_xgb}")
```

```python
# Evaluate the model
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print(f"Accuracy: {accuracy_xgb}")
```

```
Accuracy: 0.9866666666666667
```

```python
from sklearn.metrics import confusion_matrix, classification_report

cm_xgb = confusion_matrix(y_test, y_pred_xgb)
print(cm_xgb)

cs_report_xgb = classification_report(y_test, y_pred_xgb)
print(cs_report_xgb)
```

```
[[49  0  0]
 [ 2 51  0]
 [ 0  0 48]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        49
           1       1.00      0.96      0.98        53
           2       1.00      1.00      1.00        48

    accuracy                           0.99       150
   macro avg       0.99      0.99      0.99       150
weighted avg       0.99      0.99      0.99       150
```