Data Collection and Preprocessing

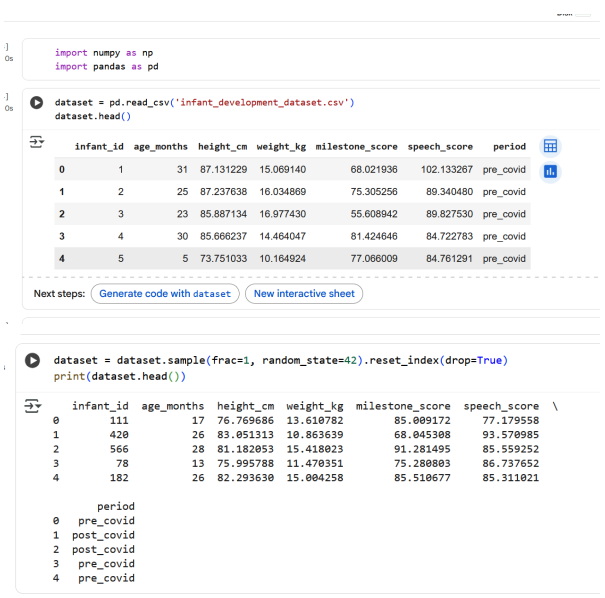
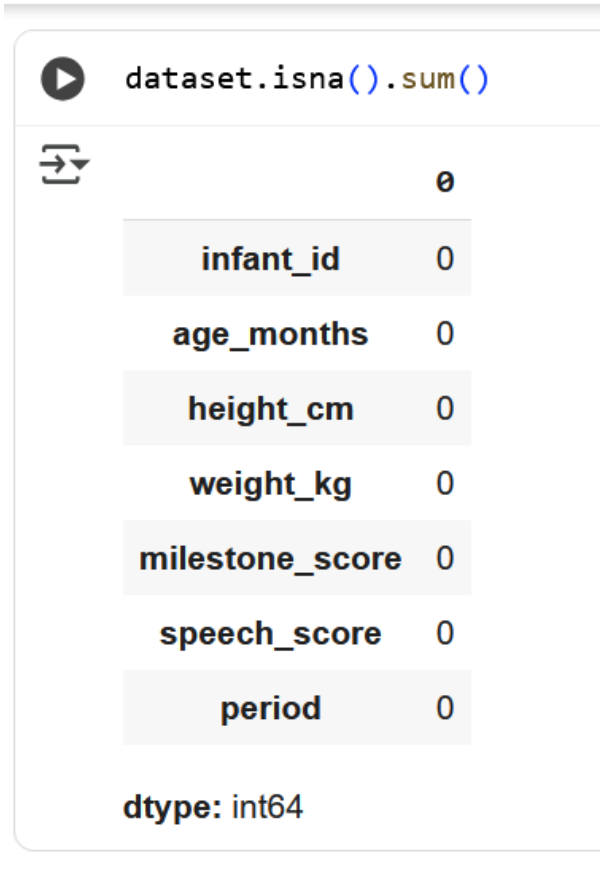
Project Name : Covid - 19 Infant Growth Analysis and Prediction

Prepare the infant development dataset for machine learning classification using TabPFN.

PREPROCESSING STEPS :

SECTION	DESCRIPTION
Data Overview	Loaded the dataset, shuffled dataset, inspected columns, and checked missing values.
Handling missing data	Filled missing numeric values (age_months, height_cm, weight_kg, speech_score, milestone_score) with their column means. For categorical column period replace missing values with the mode.
Feature & Target Split	Split dataset into features x and y
Encoding	Applied LabelEncoder to transform categorical target labels into numeric form.
Splitting dataset	Divided data into training (75%) and testing (25%) sets
Model preparation	Installed and initialized TabPFNClassifier for training.
Visualization	Generated scatter plots to compare training and testing predictions versus actual labels.

Data Preprocessing Code Screenshots :

Section	Code
Data Overview	
Handling missing data	

Feature & Target Split

```
x = dataset.iloc[:, :-1].values
x
```

```
array([[111.        ,  17.        ,  76.76968601,  13.61078245,
         85.00917188,  77.17955804],
       [420.        ,  26.        ,  83.05131333,  10.86363906,
         68.04530769,  93.57098541],
       [566.        ,  28.        ,  81.18205336,  15.41802335,
         91.28149546,  85.55925192],
       ...,
       [271.        ,  14.        ,  72.98423838,  12.75260549,
         76.82129208,  73.9939117 ],
       [436.        ,  12.        ,  74.07420025,   8.29130125,
         49.37666341,  76.08972237],
       [103.        ,  17.        ,  80.89907367,  12.23801102,
         87.68727492,  78.75725317]])
```

```
y = dataset.iloc[:, -1].values
y
```

```
       'post_covid', 'pre_covid', 'pre_covid', 'during_covid',
       'post_covid', 'pre_covid', 'during_covid', 'during_covid',
       'during_covid', 'post_covid', 'pre_covid', 'pre_covid',
       'during_covid', 'during_covid', 'during_covid', 'post_covid',
       'post_covid', 'post_covid', 'post_covid', 'post_covid',
       'during_covid', 'pre_covid', 'post_covid', 'during_covid',
       'during_covid', 'pre_covid', 'during_covid', 'pre_covid',
       'post_covid', 'during_covid', 'post_covid', 'post_covid',
       'during_covid', 'post_covid', 'post_covid', 'during_covid',
       'post_covid', 'during_covid', 'post_covid', 'post_covid',
       'post_covid', 'pre_covid', 'pre_covid', 'post_covid', 'post_covid',
       'post_covid', 'during_covid', 'during_covid', 'pre_covid',
       'post_covid', 'during_covid', 'during_covid', 'post_covid',
       'pre_covid', 'during_covid', 'pre_covid', 'post_covid',
       'pre_covid', 'post_covid', 'during_covid', 'during_covid',
       'pre_covid', 'pre_covid', 'post_covid', 'during_covid',
       'during_covid', 'during_covid', 'post_covid', 'post_covid',
       'during_covid', 'pre_covid', 'post_covid', 'during_covid',
       'post_covid', 'pre_covid', 'pre_covid', 'post_covid', 'post_covid',
       'pre_covid', 'post_covid', 'pre_covid', 'pre_covid', 'pre_covid',
       'during_covid', 'pre_covid', 'during_covid', 'pre_covid',
       'during_covid', 'pre_covid', 'pre_covid', 'pre_covid', 'pre_covid',
```

Encoding

```
#encoding missing values

dataset.fillna({
    'age_months': dataset['age_months'].mean(),
    'height_cm': dataset['height_cm'].mean(),
    'weight_kg': dataset['weight_kg'].mean(),
    'speech_score': dataset['speech_score'].mean(),
    'period': dataset['period'].mode()[0] }, inplace=True)
```

```
# Encoding categorical data

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
y
```

```
array([2, 1, 1, 2, 2, 0, 2, 1, 2, 0, 2, 2, 2, 1, 0, 0, 2, 0, 2, 2, 2, 1,
       1, 1, 0, 1, 2, 2, 1, 1, 1, 2, 2, 0, 2, 1, 2, 1, 1, 0, 0, 2, 1, 1,
       2, 2, 1, 2, 2, 1, 0, 0, 2, 1, 1, 0, 0, 2, 2, 2, 2, 2, 0, 1, 1, 2,
       1, 1, 2, 0, 0, 2, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 2, 2, 2, 0,
       0, 0, 0, 1, 2, 0, 0, 0, 0, 2, 1, 2, 2, 0, 1, 0, 2, 1, 2, 0, 0, 1,
       2, 1, 1, 1, 0, 0, 0, 2, 1, 1, 2, 2, 0, 1, 2, 2, 1, 0, 1, 1, 2, 1,
       2, 2, 1, 2, 2, 2, 0, 2, 2, 0, 2, 2, 1, 2, 0, 1, 2, 2, 1, 1, 0, 0,
       1, 0, 2, 1, 2, 2, 0, 0, 2, 1, 0, 2, 0, 0, 1, 0, 0, 0, 1, 1, 2, 2,
       1, 2, 2, 0, 2, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 2, 0, 1, 0, 1, 0,
       1, 2, 2, 1, 1, 2, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 0, 2, 0, 2, 1, 2,
       2, 1, 2, 0, 0, 1, 0, 0, 2, 0, 2, 1, 0, 1, 1, 2, 1, 0, 1, 0, 1, 0,
       1, 2, 2, 0, 1, 1, 2, 2, 2, 2, 2, 1, 2, 2, 0, 0, 2, 0, 0, 0, 1, 2,
       0, 1, 0, 2, 2, 0, 0, 0, 2, 2, 0, 1, 1, 2, 2, 1, 1, 2, 1, 1, 0, 1,
       0, 0, 2, 1, 0, 2, 0, 2, 2, 0, 1, 2, 1, 0, 2, 1, 1, 1, 2, 0, 2, 0,
       2, 2, 2, 1, 0, 1, 0, 2, 0, 2, 1, 1, 2, 0, 1, 1, 0, 0, 0, 0, 1, 2,
       0, 2, 2, 0, 0, 1, 2, 2, 0, 0, 0, 1, 0, 2, 0, 1, 1, 2, 1, 0, 2, 0,
       2, 1, 1, 2, 1, 0, 0, 1, 2, 1, 2, 2, 0, 1, 2, 0, 0, 0, 1, 2, 2, 0,
```

Splitting dataset

```
#training the data
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 0)
x_train
```

```
array([[474.        ,  23.        ,  78.97951366,  11.39366171,
         89.97368971,  72.95447563],
       [191.        ,  29.        ,  85.97125872,  14.49243952,
         77.66009333,  99.77444832],
       [222.        ,  16.        ,  68.49672067,  11.45039966,
         56.13318342,  69.78835013],
       ...,
       [514.        ,  18.        ,  80.83463022,   7.73946824,
         60.99326009,  78.1433206 ],
       [ 70.        ,  18.        ,  73.6238647 ,  11.50645298,
         92.04922922,  75.42671625],
       [274.        ,  35.        ,  83.67491912,  17.9355326 ,
         86.68249059,  73.3671546 ]])
```

Model preparation

```
!pip install tabpfn
```

```
Requirement already satisfied: scipy<2,>=1.11.1 in /usr/local/lib/python3.12/dist-packages (from tabpfn
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from tabpfn
Requirement already satisfied: einops<0.9,>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from tabp
Requirement already satisfied: huggingface-hub<1,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: pydantic<>=2.8.0 in /usr/local/lib/python3.12/dist-packages (from tabpfn)
Requirement already satisfied: pydantic-settings>=2.10.1 in /usr/local/lib/python3.12/dist-packages (fr
Collecting eval-type-backport>=0.2.2 (from tabpfn)
  Downloading eval_type_backport-0.2.2-py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from tabpfn) (
Collecting tabpfn-common-utils>=0.1.8 (from tabpfn-common-utils[telemetry-interactive]>=0.1.8->tabpfn)
  Downloading tabpfn_common_utils-0.2.1-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface-hu
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggin
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.12/dist-packages (from huggingf
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from huggingface
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from huggingface-hu
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.12/dist-packages (from huggingfac
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from hu
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas<3,
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<3,>
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<3
```

```
from tabpfn import TabPFNClassifier
```

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

```
tabpfn-v2-classifier-finetuned-zk73skhh.  29.0M/29.0M [00:01<00:00, 18.7MB/
(...): 100%                                                              s]
config.json: 100%  37.0/37.0 [00:00<00:00, 3.01kB/s]
/usr/local/lib/python3.12/dist-packages/tabpfn/classifier.py:484: UserWarning: Running on CPU with more th
Consider using a GPU or the tabpfn-client API: https://github.com/PriorLabs/tabpfn-client
  check_cpu_warning(
TabPFNClassifier
TabPFNClassifier()
```

```
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
```

Visualization

```
#vizualisation

import matplotlib.pyplot as plt

plt.scatter(x_train[:, 4],y_train,colo
plt.plot(x_train[:, 4],tabpfn.predict(x_train),color = 'blue')
plt.title('Tabpfn Model(Training)')
plt.show()

plt.scatter(x_test[:, 4],y_test,color = 'red')
plt.plot(x_test[:,4],y_pred_tabpfn,color = 'blue')
plt.title('Tabpfn Model(Testing)')
plt.show()
```

Loading...

```
#vizualisation

import matplotlib.pyplot as plt

plt.scatter(x_train[:, 4],y_train,color = 'red')
plt.plot(x_train[:, 4],xgb.predict(x_train),color = 'blue')
plt.title('XGBClassifier Model(Training)')
plt.show()

plt.scatter(x_test[:, 4],y_test,color = 'red')
plt.plot(x_test[:,4],y_pred_xgb,color = 'blue')
plt.title('XGBClassifier Model(Testing)')
plt.show()
```