<u>Naive Bayes Model Integrated with Website using FastAPI</u>

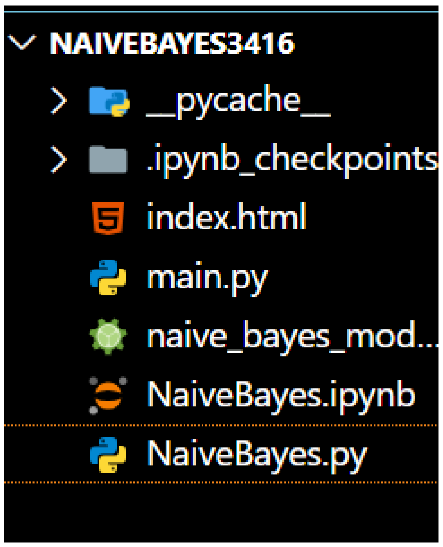
Name: Barsha Baibhabi
Roll No: 22053416

1. Overview

This project implements a Naive Bayes classifier using the Adult Income dataset from OpenML. The model predicts whether an individual earns more than $50K per year based on selected features.

List of Files

1. **`NaiveBayes.ipynb`** - Jupyter Notebook for training and saving the model.
2. **`main.py`** - FastAPI backend to serve predictions.
3. **`index.html`** - Frontend with inline CSS & JavaScript for user interaction.
4. **`naive_bayes_model.pkl`** - Saved Naive Bayes model.
5. **NaiveBayes.py** - Python script version of the Jupyter Notebook.



NAIVEBAYES3416
__pycache__
.ipynb_checkpoints
index.html
main.py
naive_bayes_mod...
NaiveBayes.ipynb
NaiveBayes.py

2. Installation & Setup

Prerequisites

Ensure you have Python installed, along with the following dependencies:

```
pip install fastapi uvicorn scikit-learn pandas numpy pydantic
```

Running the FastAPI Server

Start the FastAPI backend with:

```
uvicorn main:app --host 127.0.0.1 --port 8011 --reload
```

3. Training the Model & Generating Pickle File

The Jupyter Notebook (`naive_bayes.ipynb`) loads the dataset, preprocesses it, trains a **Gaussian Naive Bayes** model, and saves it as a pickle file.

```
PS C:\Users\KIIT\Documents\AD22053416\SVM3416> jupyter nbconvert --to script
NaiveBayes.ipynb
```

4. FastAPI Backend (`main.py`)

The backend loads the trained model and provides an API endpoint to receive feature inputs and return predictions.

```
@app.post("/predict")

async def predict(data: InputData):

    try:

        features = np.array(data.features).reshape(1, -1)

        prediction = model.predict(features)[0]

        return {"prediction": int(prediction)}

    except Exception as e:

        raise HTTPException(status_code=400, detail=str(e))

if __name__ == "__main__":

    import uvicorn

    uvicorn.run(app, host="127.0.0.1", port=8011)
```

5. Frontend (`index.html`) with Inline CSS & JavaScript

This file allows users to input data and get predictions from the FastAPI backend.

```
 async function getPrediction() {
            let input = document.getElementById("features").value;
            let featureArray = input.split(",").map(Number);

            let response = await fetch("http://127.0.0.1:8011/predict", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ features: featureArray })
            });

            let data = await response.json();
            let predictionText = data.prediction === 1 ? "Income > $50K" :
"Income ≤ $50K";
            document.getElementById("result").innerText = "Prediction: " +
predictionText;
        }
```

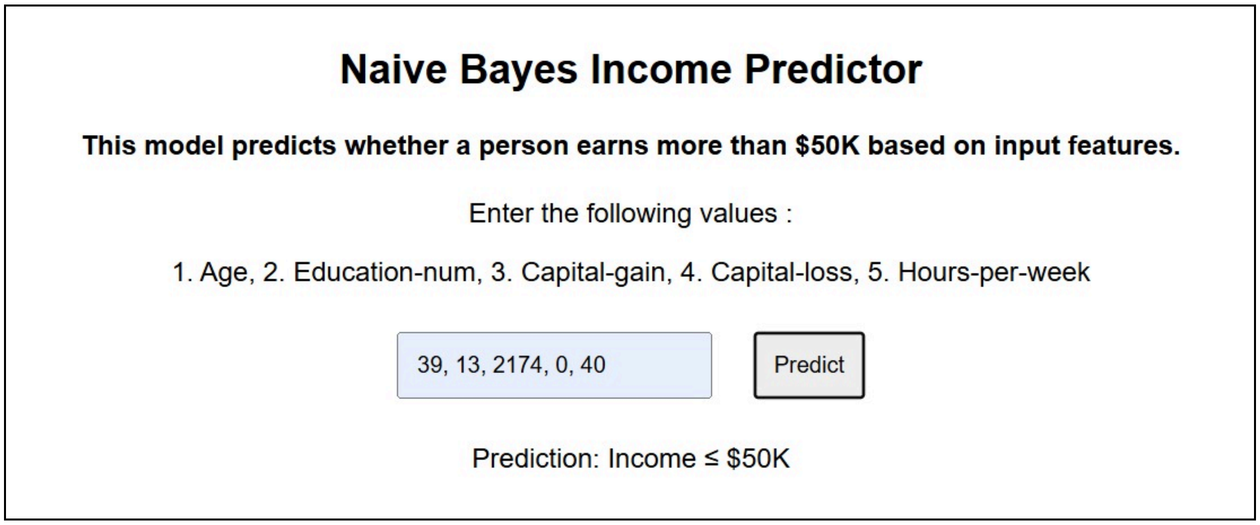
6. Testing the Integration

1. **Run FastAPI Backend:**
   `uvicorn main:app --host 127.0.0.1 --port 8011 --reload`
2. **Open `index.html` in a browser.**

3. **Enter sample values** such as:
   `39, 13, 2174, 0, 40`
4. **Click Predict** → The output will show either:

   ○ `Prediction: Income > $50K`

   ○ `Prediction: Income ≤ $50K`


Naive Bayes Income Predictor

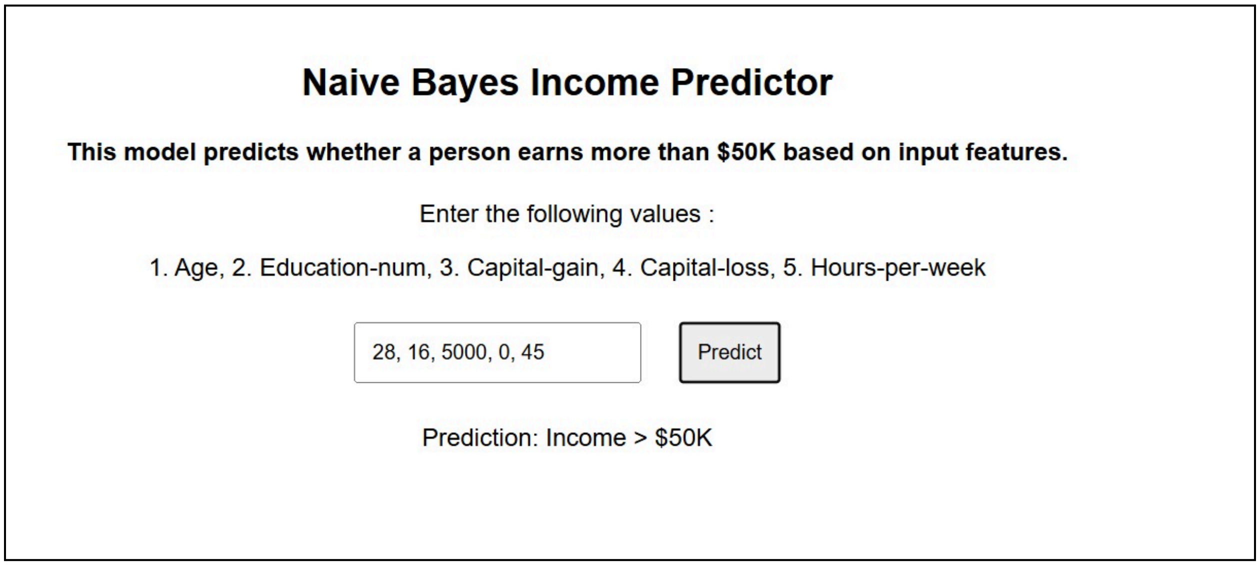
This model predicts whether a person earns more than $50K based on input features.

Enter the following values :

1. Age, 2. Education-num, 3. Capital-gain, 4. Capital-loss, 5. Hours-per-week

39, 13, 2174, 0, 40

Prediction: Income ≤ $50K


Naive Bayes Income Predictor

This model predicts whether a person earns more than $50K based on input features.

Enter the following values :

1. Age, 2. Education-num, 3. Capital-gain, 4. Capital-loss, 5. Hours-per-week

28, 16, 5000, 0, 45

Prediction: Income > $50K

7. Conclusion

This project successfully integrates a Naive Bayes classifier with a FastAPI backend and an HTML frontend for user interaction. Users can input features and receive predictions on whether they earn more than $50K. This setup can be extended to other classification problems with minor modifications.