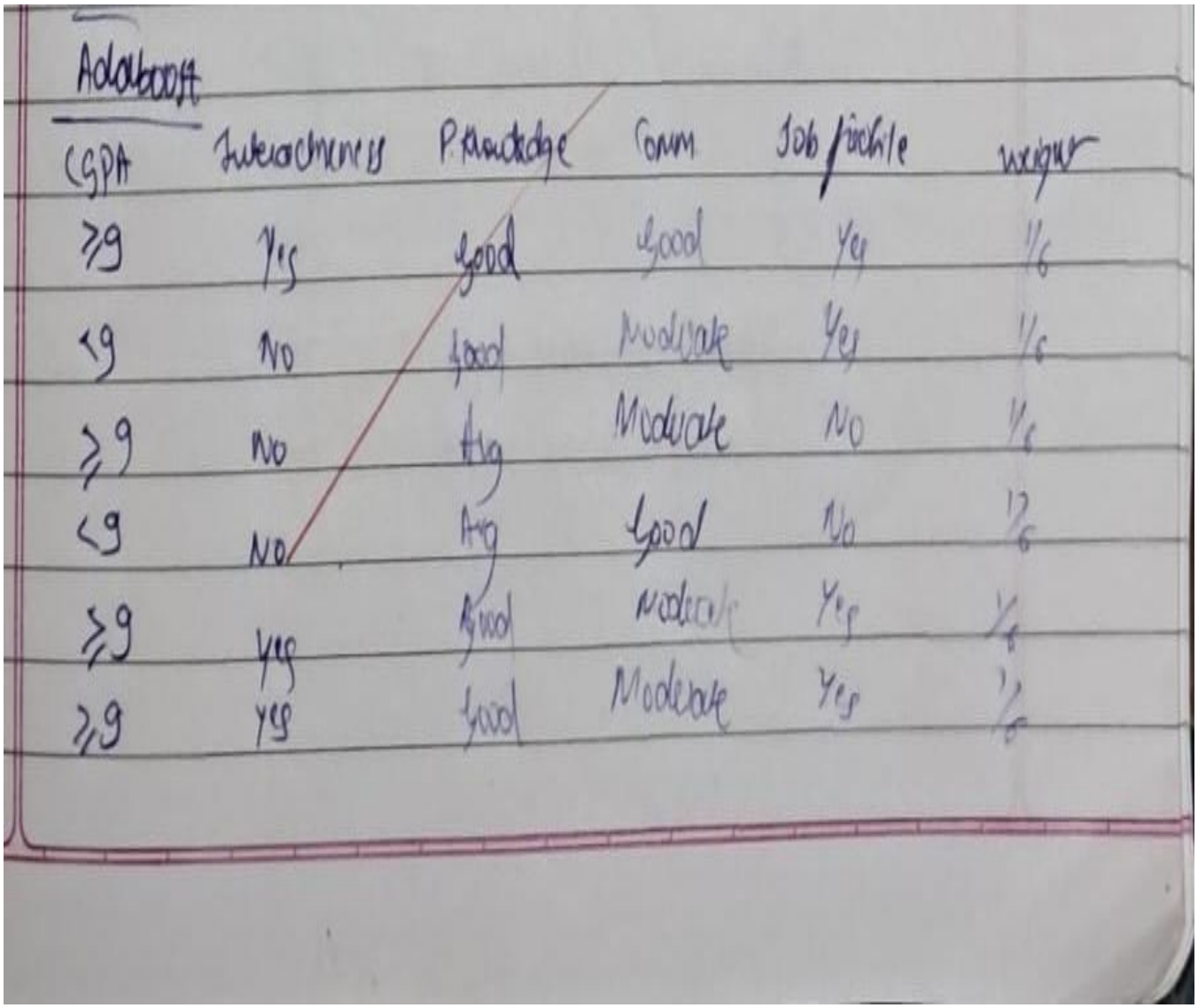
Program 9

Implement Boosting ensemble method on a given dataset

Screenshot:



Initial wt = 1/6

if CGPA ≥ 9 ⇒ Job profile = Yes

CGPA < 9 ⇒ Job profile = No

CGPA	Actual	Predicted
≥9	Yes	Yes
<9	Yes	No
≥9	No	Yes
<9	No	No
≥9	Yes	Yes
≥9	Yes	Yes

$\varepsilon = \sum w_i = \frac{2}{6} = 0.333$

$\alpha = \frac{1}{2} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) = \frac{1}{2} 0.347$

$Z_{CGPA} = \frac{1}{6} \times 4 \times e^{-0.347} + \frac{1}{6} \times 2 \times e^{0.347}$

$= 0.9438$

$Wt(d_i)_{iH} = \frac{\frac{1}{6} * e^{-0.347}}{0.9438} = 0.125$

$Wt(d_i)_{iH} = 0.2508$

But accuracy score = 83.3%

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv("/content/income.csv")

# Encode the target column (income_level)
le = LabelEncoder()
df['income_level'] = le.fit_transform(df['income_level'])  # e.g., <=50K → 0, >50K → 1

# Split features and target
X = df.drop("income_level", axis=1)
y = df["income_level"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 1 □□Train AdaBoostClassifier with n_estimators = 10
model = AdaBoostClassifier(n_estimators=10, random_state=42)
```

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy with 10 estimators: {accuracy:.4f}")


# 2 Fine-tune the model by changing n_estimators

scores = []

n_values = range(10, 101, 10)


for n in n_values:

    model = AdaBoostClassifier(n_estimators=n, random_state=42)

    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)

    scores.append(acc)

    print(f"n_estimators={n} => Accuracy: {acc:.4f}")


# 3 Find the best result

best_n = n_values[scores.index(max(scores))]

best_score = max(scores)

print(f"\n Best accuracy: {best_score:.4f} with {best_n} estimators")


# Optional: Plot the result

plt.plot(n_values, scores, marker='o')
```

```
plt.title("Accuracy vs Number of Estimators")

plt.xlabel("n_estimators (number of trees)")

plt.ylabel("Accuracy")

plt.grid(True)

plt.show()
```