NAME:-CHANDANA NH
USN:- 1SV21CS021
TEAM:- 06

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

import pandas as pd  # Import the pandas library

dia = pd.read_csv(r"/content/drive/MyDrive/Student_performance_data _
- Student_performance_data _ (1).csv")  # Now you can use pd to read
the CSV file

dia.head()
```


```
{"summary":"{\n  \"name\": \"dia\",\n  \"rows\": 2392,\n  \"fields\":
[\n    {\n      \"column\": \"StudentID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 690,\n        \"min\": 1001,\
n        \"max\": 3392,\n        \"num_unique_values\": 2392,\n
\"samples\": [\n          2005,\n          1197,\n          3343\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 1,\n        \"min\":
15,\n        \"max\": 18,\n        \"num_unique_values\": 4,\n
\"samples\": [\n          18,\n          16,\n          17\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ethnicity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 3,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          3\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"ParentalEducation\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 4,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"StudyTimeWeekly\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 5.652774235888834,\n
\"min\": 0.001056538646,\n        \"max\": 19.978094,\n
\"num_unique_values\": 2392,\n        \"samples\": [\n
0.1357634805,\n          1.989924524\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Absences\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 8,\n
```

\"min\": 0,\n   \"max\": 29,\n   \"num_unique_values\": 30,\n   \"samples\": [\n   18,\n   25\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Tutoring\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ParentalSupport\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 4,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Extracurricular\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sports\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Music\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Volunteering\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"GPA\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.9151558203270974,\n   \"min\": 0.0,\n   \"max\": 4.0,\n   \"num_unique_values\": 2371,\n   \"samples\": [\n   3.310401269,\n   3.457711726\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"GradeClass\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 4,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"dia"}

```
# Features and target variable
x = dia[['Age', 'Gender', 'Ethnicity']]  # Features
y = dia['GradeClass']  # Target variable
```

```
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

# Standardize the features (important for KNN)
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Initialize KNN classifier with a chosen number of neighbors (e.g.,
5)
k = 5  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)

knn.fit(x_train_scaled, y_train)

# ... (rest of your code)
```

```
KNeighborsClassifier()
```

```
# Predict on the test set
y_pred = knn.predict(x_test_scaled)

# Print accuracy score and classification report
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print('\nClassification Report:\n', classification_report(y_test,
y_pred))
```

```
Accuracy: 0.3695198329853862

Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00        22
           1       0.10      0.10      0.10        49
           2       0.12      0.09      0.11        85
           3       0.20      0.12      0.15        86
           4       0.50      0.65      0.56       237

    accuracy                           0.37       479
   macro avg       0.18      0.19      0.18       479
weighted avg       0.31      0.37      0.33       479


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder

import pandas as pd  # Import the pandas library

dia = pd.read_csv(r"/content/drive/MyDrive/heart.csv")  # Now you can
use pd to read the CSV file

dia.head()
```


```
{"summary":"{\n  \"name\": \"dia\",\n  \"rows\": 303,\n  \"fields\":
[\n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n
\"max\": 77,\n        \"num_unique_values\": 41,\n        \"samples\":
[\n          46,\n          66,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"cp\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 3,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"trtbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17,\n        \"min\": 94,\n
\"max\": 200,\n        \"num_unique_values\": 49,\n
\"samples\": [\n          104,\n          123\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51,\n        \"min\": 126,\n
\"max\": 564,\n        \"num_unique_values\": 152,\n
\"samples\": [\n          277,\n          169\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
```

\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   0,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"restecg\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   0,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"thalachh\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 22,\n   \"min\": 71,\n
\"max\": 202,\n   \"num_unique_values\": 91,\n
\"samples\": [\n   159,\n   152\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"exng\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   1,\n   0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"oldpeak\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 1.1610750220686348,\n   \"min\":
0.0,\n   \"max\": 6.2,\n   \"num_unique_values\": 40,\n
\"samples\": [\n   1.9,\n   3.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"slp\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   0,\n   2\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"caa\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 1,\n   \"min\": 0,\n
\"max\": 4,\n   \"num_unique_values\": 5,\n   \"samples\":
[\n   2,\n   4\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"thall\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 3,\n   \"num_unique_values\": 4,\n   \"samples\":
[\n   2,\n   0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"output\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   0,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"dia"}

```
dia.isnull().sum()
```

```
age         0
sex         0
```

```
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

```
linreg = LinearRegression()
```

```
ind = dia[['age', 'sex', 'cp', 'trtbps']]
dep = dia['output']
linreg.fit(ind,dep)
```

```
LinearRegression()
```

```
a=int(input("Enter age: "))
b=int(input("Enter sex: "))
c=int(input("Enter cp: "))
d=int(input("Enter trtbps: "))
```

```
Enter age: 63
Enter sex: 1
Enter cp: 3
Enter trtbps: 145
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
```

```
dia = pd.read_excel(r"/content/drive/MyDrive/diabetes.xlsx")
```

```
dia.head()
```


```
{"summary":"{\n  \"name\": \"dia\",\n  \"rows\": 768,\n  \"fields\":
[\n    {\n      \"column\": \"preg\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 0,\n
\"max\": 17,\n        \"num_unique_values\": 17,\n        \"samples\":
[\n          6,\n          1,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"plas\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 31,\n        \"min\": 0,\n
\"max\": 199,\n        \"num_unique_values\": 136,\n
```

\"samples\": [\n   151,\n   101,\n   112\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"pres\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 19,\n
\"min\": 0,\n   \"max\": 122,\n   \"num_unique_values\":
47,\n   \"samples\": [\n   86,\n   46,\n
85\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"skin\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 15,\n   \"min\": 0,\n   \"max\": 99,\n
\"num_unique_values\": 51,\n   \"samples\": [\n   7,\n
12,\n   48\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"insu\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 115,\n   \"min\": 0,\n   \"max\": 846,\n
\"num_unique_values\": 186,\n   \"samples\": [\n   52,\n
41,\n   183\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"mass\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 7.884160320375446,\n   \"min\": 0.0,\n   \"max\":
67.1,\n   \"num_unique_values\": 248,\n   \"samples\": [\n
19.9,\n   31.0,\n   38.1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"pedi\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.3313285950127749,\n
\"min\": 0.078,\n   \"max\": 2.42,\n
\"num_unique_values\": 517,\n   \"samples\": [\n   1.731,\
n   0.426,\n   0.138\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 11,\n   \"min\": 21,\n
\"max\": 81,\n   \"num_unique_values\": 52,\n   \"samples\":
[\n   60,\n   47,\n   72\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"class\",\n   \"properties\": {\
n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n
\"samples\": [\n   \"tested_negative\",\n
\"tested_positive\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"dia"}

```
dia.isnull().sum()
```

```
preg     0
plas     0
pres     0
skin     0
insu     0
mass     0
pedi     0
```

```
age      0
class    0
dtype: int64

x=dia[['preg','plas','pres','skin']]
y=dia['class']

LR = LogisticRegression()
LR.fit(x,y)

LogisticRegression()

l=int(input("enter preg"))
p=int(input("enter plas"))
pr=int(input("enter pres"))
s=int(input("enter skin"))

out = LR.predict([[l,p,pr,s]])
print(out)

if out==0:
  print("No Diabetes")
else:
  print("Diabetes")

enter preg6
enter plas148
enter pres72
enter skin35
['tested_positive']
Diabetes

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
LogisticRegression was fitted with feature names
  warnings.warn(

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

df =
pd.read_csv(r"/content/drive/MyDrive/manufacturing_defect_dataset.csv"
)

d=DecisionTreeClassifier()
dt.head()

{"summary":"{\n  \"name\": \"dt\",\n  \"rows\": 3240,\n  \"fields\":
[\n    {\n      \"column\": \"ProductionVolume\",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
262,\n        \"min\": 100,\n        \"max\": 999,\n
\"num_unique_values\": 862,\n        \"samples\": [\n          385,\n
192,\n          228\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ProductionCost\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 4308.051904062145,\n        \"min\":
5000.174521330492,\n        \"max\": 19993.365548756577,\n
\"num_unique_values\": 3240,\n        \"samples\": [\n
15953.737842943568,\n          18811.49912695036,\n
13607.987745156444\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SupplierQuality\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 5.759142910765757,\n        \"min\":
80.00482009370269,\n        \"max\": 99.98921362119349,\n
\"num_unique_values\": 3240,\n        \"samples\": [\n
84.20744163059672,\n          90.6010467842476,\n
83.14471330188466\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"DeliveryDelay\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 0,\n
\"max\": 5,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          1,\n          4,\n          2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DefectRate\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.3101543966950573,\n        \"min\": 0.5007098506012553,\n
\"max\": 4.998529423589416,\n        \"num_unique_values\": 3240,\n
\"samples\": [\n          4.101461867329201,\n
4.976658631866975,\n          3.4012257701663438\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"QualityScore\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
11.611750161309034,\n        \"min\": 60.01009817962248,\n
\"max\": 99.99699307332706,\n        \"num_unique_values\": 3240,\n
\"samples\": [\n          79.75077965490647,\n
88.67907604255508,\n          85.8709984432773\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"MaintenanceHours\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
6,\n        \"min\": 0,\n        \"max\": 23,\n
\"num_unique_values\": 24,\n        \"samples\": [\n          0,\n
21,\n          9\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"DowntimePercentage\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.4436844760035674,\n        \"min\":
0.0016648394255397,\n        \"max\": 4.997590657077265,\n
\"num_unique_values\": 3240,\n        \"samples\": [\n
1.799047386343167,\n          4.944017626163256,\n
```

```
2.758455230542178\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"InventoryTurnover\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2.3297913530598673,\n        \"min\":
2.0016111850744838,\n        \"max\": 9.998577322891654,\n
\"num_unique_values\": 3240,\n        \"samples\": [\n
3.9579134216566736,\n          5.485262379252041,\n
5.8250237779268925\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"StockoutRate\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.0287972301202197,\n        \"min\":
2.0517960894972377e-06,\n        \"max\": 0.0999973867646256,\n
\"num_unique_values\": 3240,\n        \"samples\": [\n
0.0330263560837115,\n          0.0447027618039343,\n
0.0653635026406835\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"WorkerProductivity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 5.723599525590738,\n        \"min\":
80.00496047416955,\n        \"max\": 99.99678580952808,\n
\"num_unique_values\": 3240,\n        \"samples\": [\n
91.94137970059217,\n          88.86923273274554,\n
99.57817198780177\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SafetyIncidents\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 0,\n
\"max\": 9,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          5,\n          7,\n          4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"EnergyConsumption\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1153.4208195800268,\n        \"min\": 1000.7201559358512,\n
\"max\": 4997.074740785263,\n        \"num_unique_values\": 3240,\n
\"samples\": [\n          1416.715355037993,\n
3289.8774169668563,\n          2264.27115985648\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"EnergyEfficiency\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.1163996994971205,\n        \"min\": 0.1002379032170629,\n
\"max\": 0.4994998046638267,\n        \"num_unique_values\": 3240,\n
\"samples\": [\n          0.1955823399781935,\n
0.1674234250881216,\n          0.1428485726433269\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"AdditiveProcessTime\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.598211986310066,\n        \"min\": 1.0001506263214126,\n
\"max\": 9.99974932631184,\n        \"num_unique_values\": 3240,\n
\"samples\": [\n          9.476782214498312,\n
6.645386486056385,\n          3.672113025027017\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"AdditiveMaterialCost\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
116.3799050389571,\n        \"min\": 100.21113747433294,\n
\"max\": 499.98278174750806,\n        \"num_unique_values\": 3240,\n
\"samples\": [\n          290.0917396850834,\n
199.8182231770064,\n          245.053103411413\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DefectStatus\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"dt"}
```

```
dt.isnull().sum()
```

```
ProductionVolume         0
ProductionCost           0
SupplierQuality          0
DeliveryDelay            0
DefectRate               0
QualityScore             0
MaintenanceHours         0
DowntimePercentage       0
InventoryTurnover        0
StockoutRate             0
WorkerProductivity       0
SafetyIncidents          0
EnergyConsumption        0
EnergyEfficiency         0
AdditiveProcessTime      0
AdditiveMaterialCost     0
DefectStatus             0
dtype: int64
```

```
x=dt[['ProductionVolume','ProductionCost','SupplierQuality','DefectRat
e','QualityScore',]]
y=dt['DefectStatus']
d.fit(x,y)
```

```
DecisionTreeClassifier()
```

```
columns_to_drop =
['DeliveryDelay','MaintenanceHours','DowntimePercentage','InventoryTur
nover']
dt = dt.drop(columns_to_drop, axis=1) # Use axis=1 to drop columns
```

```
d.predict([[100,200,65,30,52]]) # Remove the extra feature from the
input
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
DecisionTreeClassifier was fitted with feature names
  warnings.warn(

array([1])

l1=int(input("enter prduction volume"))
l2=float(input("production cost"))
l3=float(input("supplier quality"))
l4=float(input("defect rate"))
l5=float(input("quality score"))
out=d.predict([[l1,l2,l3,l4,l5]])
if out==True:
  print("positive")
else:
  print("negative")

enter prduction volume335
production cost13258.368
supplier quality53.65
defect rate1.003
quality score86.36
negative

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
DecisionTreeClassifier was fitted with feature names
  warnings.warn(

import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier

df = pd.read_csv(r"/content/drive/MyDrive/placement-dataset -
placement-dataset (1).csv")

df.head()

{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 100,\n  \"fields\": [\
n    {\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 29,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 100,\n
\"samples\": [\n          83,\n          53,\n          70\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cgpa\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.1436336737775692,\n        \"min\": 3.3,\n        \"max\":
8.5,\n        \"num_unique_values\": 39,\n        \"samples\": [\n
6.3,\n          4.4,\n          5.8\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"iq\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 39,\n        \"min\": 37,\n
\"max\": 233,\n        \"num_unique_values\": 71,\n
\"samples\": [\n          91,\n          123,\n          64\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"placement\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

```
feature = df[['cgpa','iq']]
Target = df['placement']
```

```
RF = RandomForestClassifier()
RF.fit(feature,Target)
```

```
RandomForestClassifier()
```

```
RandomForestClassifier()
```

```
RandomForestClassifier()
```

```
l=int(input("enter cgpa:"))
cl=int(input("enter iq:"))
```

```
pred = RF.predict([[l,cl]])
```

```
if pred[0] ==1:
  print("placement<3.0")
elif pred[0] ==2:
  print("placement>3.0")
```

```
enter cgpa:68
enter iq:123
placement<3.0
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but
RandomForestClassifier was fitted with feature names
  warnings.warn(
```