```python
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/adult.csv")  # Now
you can use pd to read the CSV file

dia.head()
```


```
{"summary":"{\n  \"name\": \"dia\",\n  \"rows\": 48842,\n  \"fields\":
[\n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 13,\n        \"min\": 17,\n
\"max\": 90,\n        \"num_unique_values\": 74,\n        \"samples\":
[\n          18,\n          74,\n          40\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"workclass\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 9,\n        \"samples\": [\n
\"Without-pay\",\n          \"Local-gov\",\n          \"State-gov\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"fnlwgt\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 105604,\n
\"min\": 12285,\n        \"max\": 1490400,\n
\"num_unique_values\": 28523,\n        \"samples\": [\n
171041,\n          20296,\n          263896\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"education\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 16,\n        \"samples\": [\n
\"11th\",\n          \"HS-grad\",\n          \"Prof-school\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"educational-num\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 1,\n
\"max\": 16,\n        \"num_unique_values\": 16,\n        \"samples\":
[\n          7,\n          9,\n          15\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"marital-status\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 7,\n        \"samples\": [\n          \"Never-
married\",\n          \"Married-civ-spouse\",\n          \"Married-
spouse-absent\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"occupation\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 15,\n
```

```
\"samples\": [\n          \"Tech-support\",\n          \"Priv-house-
serv\",\n          \"Machine-op-inspct\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"relationship\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n          \"Own-
child\",\n          \"Husband\",\n          \"Other-relative\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"race\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
5,\n        \"samples\": [\n          \"White\",\n          \"Amer-
Indian-Eskimo\",\n          \"Asian-Pac-Islander\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"gender\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
2,\n        \"samples\": [\n          \"Female\",\n          \"Male\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"capital-gain\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 7452,\n        \"min\": 0,\n
\"max\": 99999,\n        \"num_unique_values\": 123,\n
\"samples\": [\n          4064,\n          4787\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"capital-loss\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
403,\n        \"min\": 0,\n        \"max\": 4356,\n
\"num_unique_values\": 99,\n        \"samples\": [\n          2238,\n
1564\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"hours-per-week\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 12,\n        \"min\": 1,\n
\"max\": 99,\n        \"num_unique_values\": 96,\n        \"samples\":
[\n          9,\n          11\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"native-country\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 42,\n
\"samples\": [\n          \"Canada\",\n          \"Vietnam\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"income\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \">50K\",\n          \"<=50K\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"dia"}

# Assuming dia is your dataframe
x = dia[['Age']]  # Features - Only include numerical columns
y = dia['GradeClass']  # Target variable

# Split the data into training and testing sets
```

```python
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)
from sklearn.neighbors import KNeighborsClassifier # Import the
KNeighborsClassifier class
k = 5  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)

# Fit the model
knn.fit(x_train_scaled, y_train)
```

```
KNeighborsClassifier()
```

```python
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

# Example DataFrame creation (replace with your actual data loading
code)
data = {
    'Age': [25, 30, 35, 40, 45],
    'Gender': [0, 1, 0, 1, 0],  # Assuming encoded categorical
variables (0 and 1 for example)
    'Ethnicity': [0, 1, 2, 3, 0],  # Assuming encoded categorical
variables (0, 1, 2, 3 for example)
    'GradeClass': [1, 2, 1, 3, 2]  # Example target variable
}
dia = pd.DataFrame(data)

# 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
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

```python
# Initialize KNN classifier
k = min(5, x_train_scaled.shape[0])  # Number of neighbors - adjust based on the size of your training data
knn = KNeighborsClassifier(n_neighbors=k)

# Fit the model
knn.fit(x_train_scaled, y_train)

# 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: 0.0

Classification Report:
               precision    recall  f1-score   support

           1       0.00      0.00      0.00       0.0
           2       0.00      0.00      0.00       1.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0


/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/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true 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/_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/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true 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/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

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

import pandas as pd

# Read CSV file
dia = pd.read_csv(r"/content/drive/MyDrive/archive (9)/Diabetes_prediction.csv")

dia.head()
```


{"summary":"{\n \"name\": \"dia\",\n \"rows\": 1000,\n \"fields\": [\n   {\n   \"column\": \"Pregnancies\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 8,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   1,\n   5,\n   2\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Glucose\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 19.47073016206889,\n   \"min\": 30.571402161232346,\n   \"max\": 161.23893930812437,\n   \"num_unique_values\": 1000,\n   \"samples\": [\n   96.60663682437556,\n   106.72142335851338,\n   96.86515848435448\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"BloodPressure\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 13.882017449176828,\n   \"min\": 31.40148707615002,\n   \"max\": 110.72371460214974,\n   \"num_unique_values\": 1000,\n   \"samples\": [\n   76.46315367622508,\n   64.70790868788801,\n   61.79850706275893\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SkinThickness\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1.1738067350468842,\n   \"min\": 19.369987239303853,\n   \"max\": 26.917654051162653,\n   \"num_unique_values\": 1000,\n   \"samples\": [\n   25.019103341870355,\n   24.91318184958465,\n   25.637893190161734\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Insulin\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
74.87273276449727,\n   \"min\": -165.3100326101158,\n
\"max\": 317.701851723486,\n   \"num_unique_values\": 1000,\n
\"samples\": [\n   149.55308374851455,\n
144.28304811266074,\n   116.48511037821226\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"BMI\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 3.690223424365158,\n
\"min\": 13.548817852644664,\n   \"max\": 36.324597869685,\n
\"num_unique_values\": 1000,\n   \"samples\": [\n
24.218736681725392,\n   23.801312548795064,\n
22.9110603038601\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DiabetesPedigreeFunction\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.19933443406994794,\n
\"min\": 0.10003697885260 77,\n   \"max\": 0.7996536327959616,\n
\"num_unique_values\": 1000,\n   \"samples\": [\n
0.7326370829793747,\n   0.2484194953515602,\n
0.3541928131816235\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 14.465398004198562,\n   \"min\": -0.979803578151646,\n
\"max\": 90.5737824472806,\n   \"num_unique_values\": 1000,\n
\"samples\": [\n   17.028898711185644,\n
20.705238388105126,\n   45.54948444880257\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Diagnosis\",\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}","type":"dataframe","variable_name":"dia"}

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

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Diagnosis                   0
dtype: int64
```

```python
print(dia.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Diagnosis'],
      dtype='object')
```

```
x = dia[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness']]
# Remove the extra space after 'Glucose'
y = dia['Diagnosis']

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

```
LogisticRegression()
```

```
l=int(input("enter Pregnancies "))
p=int(input("enter Glucose "))
pr=int(input("enter BloodPressure"))
s=int(input("enter SkinThickness "))

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

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

```
enter Pregnancies 1
enter Glucose 2
enter BloodPressure3
enter SkinThickness 4
[0]
No 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 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/archive (8)/Samsung
Dataset.csv")  # Now you can use pd to read the CSV file

dia.head()
```

{"summary":"{\n \"name\": \"dia\",\n \"rows\": 6127,\n \"fields\": [\n   {\n   \"column\": \"Date\",\n   \"properties\": {\n   \"dtype\": \"object\",\n   \"num_unique_values\": 6127,\n   \"samples\": [\n   \"2005-03-17\",\n   \"2022-07-18\",\n   \"2017-04-27\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Open\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 22589.409269155833,\n   \"min\": 2540.0,\n   \"max\": 90300.0,\n   \"num_unique_values\": 2127,\n   \"samples\": [\n   7400.0,\n   76000.0,\n   46150.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"High\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 22764.800971727953,\n   \"min\": 2760.0,\n   \"max\": 96800.0,\n   \"num_unique_values\": 2209,\n   \"samples\": [\n   16440.0,\n   42120.0,\n   32980.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Low\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 22394.68127577785,\n   \"min\": 2420.0,\n   \"max\": 89500.0,\n   \"num_unique_values\": 2234,\n   \"samples\": [\n   21060.0,\n   8830.0,\n   9020.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Close\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 22567.361619100076,\n   \"min\": 2730.0,\n   \"max\": 91000.0,\n   \"num_unique_values\": 2185,\n   \"samples\": [\n   7270.0,\n   6130.0,\n   26060.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Adj Close\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 22041.30289886259,\n   \"min\": 1988.168701,\n   \"max\": 85300.0,\n   \"num_unique_values\": 3924,\n   \"samples\": [\n   55525.496094,\n   7763.32666,\n   40948.972656\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Volume\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15058880,\n   \"min\": 0,\n   \"max\": 164215000,\n   \"num_unique_values\": 5884,\n   \"samples\": [\n   15680447,\n   13532700,\n   53740000\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"dia"}

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

```
Date      0
Open      0
High      0
Low       0
Close     0
```

```
Adj Close    0
Volume       0
dtype: int64

linreg = LinearRegression()

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

<ipython-input-26-bc6306ae6027>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  ind['Date'] = ind['Date'].apply(lambda x: x.toordinal())

LinearRegression()

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

df =
pd.read_csv(r"/content/drive/MyDrive/online_sas/online_review.csv")
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2304,\n  \"fields\":
[\n    {\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 665,\n        \"min\": 0,\n
\"max\": 2303,\n        \"num_unique_values\": 2304,\n
\"samples\": [\n          1640,\n          508,\n          1422\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Product_name\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 231,\n        \"samples\": [\n          \"LG 24
inch Full HD LED Backlit IPS Panel Monitor (24MP400)\\u00a0\\
u00a0(Response Time: 5 ms)\",\n          \"LG 260 L Frost Free Double
Door Top Mount 3 Star Convertible Refrigerator\\u00a0\\u00a0(Dazzle
Steel, GL-S292RDSX)\",\n          \"HP Ryzen 3 Dual Core 3250U - (8
GB/256 GB SSD/Windows 10 Home) 15s-GY0501AU Thin and Light Laptop\\
u00a0\\u00a0(15.6 inch, Natural Silver, 1.69 kg, With MS Office)\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Review\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
1358,\n        \"samples\": [\n          \"Im statisfied .. valueble
money\",\n          \"Nice product nice design but not big actually
same 7.5 kg size...\",\n          \"awesom ips led monitor\"\
n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Rating\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 1,\n   \"max\": 5,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   4,\n
1,\n   3\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
import pandas as pd

# Example dataframe structure
data = {
    'Unnamed: 0': [1, 2, 3],
    'Product_name': ['Product A', 'Product B', 'Product C'],
    'Review': ['Good', 'Bad', 'Neutral'],
    'Rating': [4.5, 2.3, 3.0],
    'Sentiment': ['Positive', 'Negative', 'Neutral']
}

df = pd.DataFrame(data)


feature = df[['Unnamed: 0', 'Review']]  # Selecting specific columns
as features
Target = df['Rating']  # Selecting the target variable

# One-hot encode categorical variables in 'Review'
# Using pandas get_dummies for simplicity
feature = pd.get_dummies(feature, columns=['Review'])

# Create RandomForestClassifier instance
RF = RandomForestClassifier(n_estimators=10)
```

```
RandomForestClassifier()
```

RandomForestClassifier()

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

dt= pd.read_csv("/content/drive/MyDrive/archive (12)/Car Data.csv")


d=DecisionTreeClassifier()
dt.head()
```

```
{"summary":"{\n  \"name\": \"dt\",\n  \"rows\": 2000,\n  \"fields\": [\n    {\n      \"column\": \"Car ID\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 577,\n        \"min\": 1,\n        \"max\": 2000,\n        \"num_unique_values\": 2000,\n        \"samples\": [\n          1861,\n          354,\n          1334\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Brand\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"Honda\",\n          \"Hyundai\",\n          \"Ford\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Model\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 68,\n        \"samples\": [\n          \"Rav10\",\n          \"Pilot\",\n          \"Elantra\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 2015,\n        \"max\": 2020,\n        \"num_unique_values\": 6,\n        \"samples\": [\n          2018,\n          2019,\n          2015\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Color\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 7,\n        \"samples\": [\n          \"White\",\n          \"Blue\",\n          \"Gray\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Mileage\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 11016,\n        \"min\": 25000,\n        \"max\": 70000,\n        \"num_unique_values\": 10,\n        \"samples\": [\n          50000,\n          35000,\n          25000\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Price\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 4777,\n        \"min\": 12000,\n        \"max\": 29000,\n        \"num_unique_values\": 17,\n        \"samples\": [\n          18000,\n          16000,\n          19000\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Location\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 10,\n        \"samples\": [\n          \"Houston\",\n          \"New York\",\n          \"Dallas\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"dt"}
```


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

```
Car ID      0
Brand       0
Model       0
Year        0
```

```
Color       0
Mileage     0
Price       0
Location    0
dtype: int64
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.preprocessing import LabelEncoder # Import LabelEncoder
for encoding categorical features

dt= pd.read_csv("/content/drive/MyDrive/archive (12)/Car Data.csv")

d=DecisionTreeClassifier()
dt.head()

# ... (rest of your code)

x=dt[['Car ID','Brand','Model','Year','Color',]]
y=dt['Location']

# Initialize LabelEncoder
le = LabelEncoder()

# Iterate through columns and encode categorical features
for col in x.columns:
    if x[col].dtype == 'object':  # Check if the column is of object
(string) type
        x[col] = le.fit_transform(x[col])  # Encode the categorical
values

d.fit(x,y)  # Now fit the model with encoded features
```

```
<ipython-input-13-e6278cf639b3>:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  x[col] = le.fit_transform(x[col])  # Encode the categorical values
<ipython-input-13-e6278cf639b3>:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

```
  x[col] = le.fit_transform(x[col])  # Encode the categorical values
<ipython-input-13-e6278cf639b3>:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  x[col] = le.fit_transform(x[col])  # Encode the categorical values
```

```
DecisionTreeClassifier()
```

```
colums_to_drop = ['Mileage','Price']
dt.drop(colums_to_drop,axis=1)
```


```
{"summary":"{\n  \"name\": \"dt\",\n  \"rows\": 2000,\n  \"fields\":
[\n    {\n      \"column\": \"Car ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 577,\n        \"min\": 1,\n
\"max\": 2000,\n        \"num_unique_values\": 2000,\n
\"samples\": [\n          1861,\n          354,\n          1334\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Brand\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
5,\n        \"samples\": [\n          \"Honda\",\n
\"Hyundai\",\n          \"Ford\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Model\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 68,\
n        \"samples\": [\n          \"Rav10\",\n          \"Pilot\",\n
\"Elantra\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 2015,\n        \"max\": 2020,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          2018,\n
2019,\n          2015\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Color\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 7,\n        \"samples\": [\n
\"White\",\n          \"Blue\",\n          \"Gray\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Location\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
10,\n        \"samples\": [\n          \"Houston\",\n          \"New
York\",\n          \"Dallas\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


```
d.predict([[100,200,65,30.52,45.3]])
```

```
/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(['Los Angeles'], dtype=object)
```

```python
l1=int(input("enter car id")) # Ask for Car ID
l2=int(input("enter brand"))
l3=int(input("enter model"))
l4=int(input("enter year"))
l5=int(input("enter color"))
out = d.predict([[l1,l2,l3,l4,l5]]) # Predict using 5 features
if out==True:
  print("sold")
else:
  print("not sold")
```

```
enter car id1
enter brand3
enter model5
enter year7
enter color8
not sold
```

```
/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(
```