```
import numpy as np
import pandas as pd
a=pd.read_csv("/content/train.csv")
print(a)
print(a.head())
print(a.columns)
target_variable = 'price_range'
features = a.columns[a.columns != target_variable]
print('Target Variable:', target_variable)
print('Features:',features)
```

```
      2       0.9        145        5  ...       1263      1716  2603    11     2
      3       0.8        131        6  ...       1216      1786  2769    16     8
      4       0.6        141        2  ...       1208      1212  1411     8     2
      ...     ...        ...      ...  ...        ...       ...   ...   ...   ...
      1995    0.8        106        6  ...       1222      1890   668    13     4
      1996    0.2        187        4  ...        915      1965  2032    11    10
      1997    0.7        108        8  ...        868      1632  3057     9     1
      1998    0.1        145        5  ...        336       670   869    18    10
      1999    0.9        168        6  ...        483       754  3919    19     4

            talk_time  three_g  touch_screen  wifi  price_range
      0            19        0             0     1            1
      1             7        1             1     0            2
      2             9        1             1     0            2
      3            11        1             0     0            2
      4            15        1             1     0            1
      ...         ...      ...           ...   ...          ...
      1995         19        1             1     0            0
      1996         16        1             1     1            2
      1997          5        1             1     0            3
      1998         19        1             1     1            0
      1999          2        1             1     1            3

      [2000 rows x 21 columns]
         battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  m_dep  \
      0            842     0          2.2         0   1       0           7    0.6
```

```
3        1             0     0            2
4        1             1     0            1

[5 rows x 21 columns]
Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
       'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
       'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
       'touch_screen', 'wifi', 'price_range'],
      dtype='object')
Target Variable: price_range
Features: Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
       'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
       'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
       'touch_screen', 'wifi'],
      dtype='object')
```

```
y=a['price_range']
y
```

```
0       1
1       2
2       2
3       2
4       1
       ..
1995    0
1996    2
1997    3
1998    0
1999    3
Name: price_range, Length: 2000, dtype: int64
```

```
X=a.drop('price_range',axis=1)
X
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_
0	842	0	2.2	0	1	0	7	0.6	1
1	1021	1	0.5	1	0	1	53	0.7	1
2	563	1	0.5	1	2	1	41	0.9	1
3	615	1	2.5	0	0	0	10	0.8	1
4	1821	1	1.2	0	13	1	44	0.6	1
...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	1
1996	1965	1	2.6	1	0	0	39	0.2	1
1997	1911	0	0.9	1	1	1	36	0.7	1
1998	1512	0	0.9	0	4	1	46	0.1	1
1999	510	1	2.0	1	5	1	45	0.9	1

2000 rows × 20 columns

Next steps: View recommended plots

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("\nScaled data:")
print(pd.DataFrame(X_train_scaled, columns=X_train.columns).head())
```

```
Scaled data:
   battery_power  blue  clock_speed  dual_sim        fc  four_g  int_memory  \
0       0.577540   0.0         0.04       0.0  0.000000     1.0    0.467742
1       0.227273   1.0         0.00       1.0  0.421053     0.0    0.596774
2       0.315508   0.0         0.00       1.0  0.263158     0.0    0.822581
3       0.017380   0.0         0.08       1.0  0.210526     0.0    0.822581
4       0.077540   1.0         0.72       0.0  0.000000     0.0    0.483871

      m_dep  mobile_wt   n_cores    pc  px_height  px_width       ram  \
0  0.222222   0.375000  0.428571  0.00   0.149490  0.313752  0.278610
1  0.000000   0.991667  0.714286  0.65   0.278571  0.218959  0.717647
2  0.444444   0.966667  0.571429  0.35   0.053571  0.058745  0.428610
3  0.888889   0.400000  0.000000  0.55   0.034184  0.950601  0.279144
4  0.444444   0.991667  0.857143  1.00   0.391837  0.249666  0.522727
```

	sc_h	sc_w	talk_time	three_g	touch_screen	wifi
0	0.642857	0.388889	0.611111	1.0	0.0	1.0
1	0.428571	0.277778	0.777778	0.0	1.0	1.0
2	0.571429	0.555556	0.777778	0.0	1.0	0.0
3	0.571429	0.388889	0.611111	0.0	0.0	0.0
4	0.928571	0.722222	0.277778	0.0	0.0	0.0

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.605
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.73      0.78       166
           1       0.49      0.56      0.52       136
           2       0.40      0.38      0.39       141
           3       0.66      0.71      0.69       157

    accuracy                           0.60       600
   macro avg       0.60      0.60      0.60       600
weighted avg       0.61      0.60      0.61       600
```

```
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[122  40   4   0]
 [ 23  76  32   5]
```

```
[  0  36  53  52]
[  0   2  43 112]]
```