```
'''
2) How to find given fruit.Csv To find feature and target values
and can use algorithms Logistics regression, decision tree, svm, random forest regression model can implement it??
'''

import pandas as pd

# Load data

df = pd.read_csv("/content/fruit.csv")
print(df)
```

```
    fruit_label fruit_name     fruit_subtype  mass  width  height  color_score
0             1      apple      granny_smith   192    8.4     7.3         0.55
1             1      apple      granny_smith   180    8.0     6.8         0.59
2             1      apple      granny_smith   176    7.4     7.2         0.60
3             2   mandarin          mandarin    86    6.2     4.7         0.80
4             2   mandarin          mandarin    84    6.0     4.6         0.79
5             2   mandarin          mandarin    80    5.8     4.3         0.77
6             2   mandarin          mandarin    80    5.9     4.3         0.81
7             2   mandarin          mandarin    76    5.8     4.0         0.81
8             1      apple          braeburn   178    7.1     7.8         0.92
9             1      apple          braeburn   172    7.4     7.0         0.89
10            1      apple          braeburn   166    6.9     7.3         0.93
11            1      apple          braeburn   172    7.1     7.6         0.92
12            1      apple          braeburn   154    7.0     7.1         0.88
13            1      apple  golden_delicious   164    7.3     7.7         0.70
14            1      apple  golden_delicious   152    7.6     7.3         0.69
15            1      apple  golden_delicious   156    7.7     7.1         0.69
16            1      apple  golden_delicious   156    7.6     7.5         0.67
17            1      apple  golden_delicious   168    7.5     7.6         0.73
18            1      apple       cripps_pink   162    7.5     7.1         0.83
19            1      apple       cripps_pink   162    7.4     7.2         0.85
20            1      apple       cripps_pink   160    7.5     7.5         0.86
21            1      apple       cripps_pink   156    7.4     7.4         0.84
22            1      apple       cripps_pink   140    7.3     7.1         0.87
23            1      apple       cripps_pink   170    7.6     7.9         0.88
24            3     orange     spanish_jumbo   342    9.0     9.4         0.75
25            3     orange     spanish_jumbo   356    9.2     9.2         0.75
26            3     orange     spanish_jumbo   362    9.6     9.2         0.74
27            3     orange  selected_seconds   204    7.5     9.2         0.77
28            3     orange  selected_seconds   140    6.7     7.1         0.72
29            3     orange  selected_seconds   160    7.0     7.4         0.81
30            3     orange  selected_seconds   158    7.1     7.5         0.79
31            3     orange  selected_seconds   210    7.8     8.0         0.82
32            3     orange  selected_seconds   164    7.2     7.0         0.80
33            3     orange       urkey_navel   190    7.5     8.1         0.74
34            3     orange      turkey_navel   142    7.6     7.8         0.75
35            3     orange      turkey_navel   150    7.1     7.9         0.75
36            3     orange      turkey_navel   160    7.1     7.6         0.76
37            3     orange      turkey_navel   154    7.3     7.3         0.79
38            3     orange      turkey_navel   158    7.2     7.8         0.77
39            3     orange      turkey_navel   144    6.8     7.4         0.75
40            3     orange      turkey_navel   154    7.1     7.5         0.78
41            3     orange      turkey_navel   180    7.6     8.2         0.79
42            3     orange      turkey_navel   154    7.2     7.2         0.82
43            4      lemon    spanish_belsan   194    7.2    10.3         0.70
44            4      lemon    spanish_belsan   200    7.3    10.5         0.72
45            4      lemon    spanish_belsan   186    7.2     9.2         0.72
46            4      lemon    spanish_belsan   216    7.3    10.2         0.71
47            4      lemon    spanish_belsan   196    7.3     9.7         0.72
48            4      lemon    spanish_belsan   174    7.3    10.1         0.72
49            4      lemon           unknown   132    5.8     8.7         0.73
50            4      lemon           unknown   130    6.0     8.2         0.71
51            4      lemon           unknown   116    6.0     7.5         0.72
52            4      lemon           unknown   118    5.9     8.0         0.72
53            4      lemon           unknown   120    6.0     8.4         0.74
54            4      lemon           unknown   116    6.1     8.5         0.71
55            4      lemon           unknown   116    6.3     7.7         0.72
56            4      lemon           unknown   116    5.9     8.1         0.73
```

```
print(df.head())
```

```
   fruit_label fruit_name  fruit_subtype  mass  width  height  color_score
0            1      apple   granny_smith   192    8.4     7.3         0.55
1            1      apple   granny_smith   180    8.0     6.8         0.59
2            1      apple   granny_smith   176    7.4     7.2         0.60
3            2   mandarin       mandarin    86    6.2     4.7         0.80
4            2   mandarin       mandarin    84    6.0     4.6         0.79
```

```
print(df.columns)
```

```
Index(['fruit_label', 'fruit_name', 'fruit_subtype', 'mass', 'width', 'height',
       'color_score'],
```

```
      dtype='object')
```

```
# Define feature "x" and target "y"

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['fruit_label'] = le.fit_transform(df['fruit_label'])

# Features
X = df[['mass', 'width', 'height', 'color_score']]  # numeric features
y = df['fruit_label']

# print(X,y)


# Train-Test Split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=42)

#print(X_train)
#print(y_train)


# ALGORITHMS:

# a) Logistic algorithm:


from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
print("LR Accuracy:", model.score(X_test, y_test))
```

```
LR Accuracy: 0.75
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=
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(
```

```
# b) Decision Tree

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
print("DT Accuracy:", model.score(X_test, y_test))
```

```
DT Accuracy: 0.8333333333333334
```

```
# c) SVM

from sklearn.svm import SVC

model = SVC()
model.fit(X_train, y_train)
print("SVM Accuracy:", model.score(X_test, y_test))
```

```
SVM Accuracy: 0.5
```

+ Code + Text

```
# d) Random Forest

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
print("RF Accuracy:", model.score(X_test, y_test))
```

```
RF Accuracy: 1.0
```

```
# 5. Predict New Data

# Example: Predict fruit with new values
sample = [[150, 7.0, 6.5, 0.55]]  # [mass, width, height, color_score]
```

```
pred = model.predict(sample)
print("Predicted Fruit:", le.inverse_transform(pred)[0])
```

```
Predicted Fruit: 0
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Ranc
  warnings.warn(
```