IMAGE CLASSIFICATION USING DECISION TREE

Decision trees are essentially binary trees that recursively split the data set until we are left with pure leaf nodes. The Decision nodes are the conditions to split the dataset and the leaf nodes help to decide the class of the dataset.

A Decision Tree is a Supervised Learning Algorithm that works for both the cases of Classification and Regression (DecisionTreeClassifier() and DecisionTreeRegressor() in python.)

https://medium.com/@theclickreader/decision-tree-regression-explained-with-implementation-in-python-1e6e48aa7a47 - Read how a Decision Tree Regression algorithm works here.

Image Classification is a Classification Problem that falls under Computer Vision. It is the process of categorising and labelling pixels in an image using specific rules.

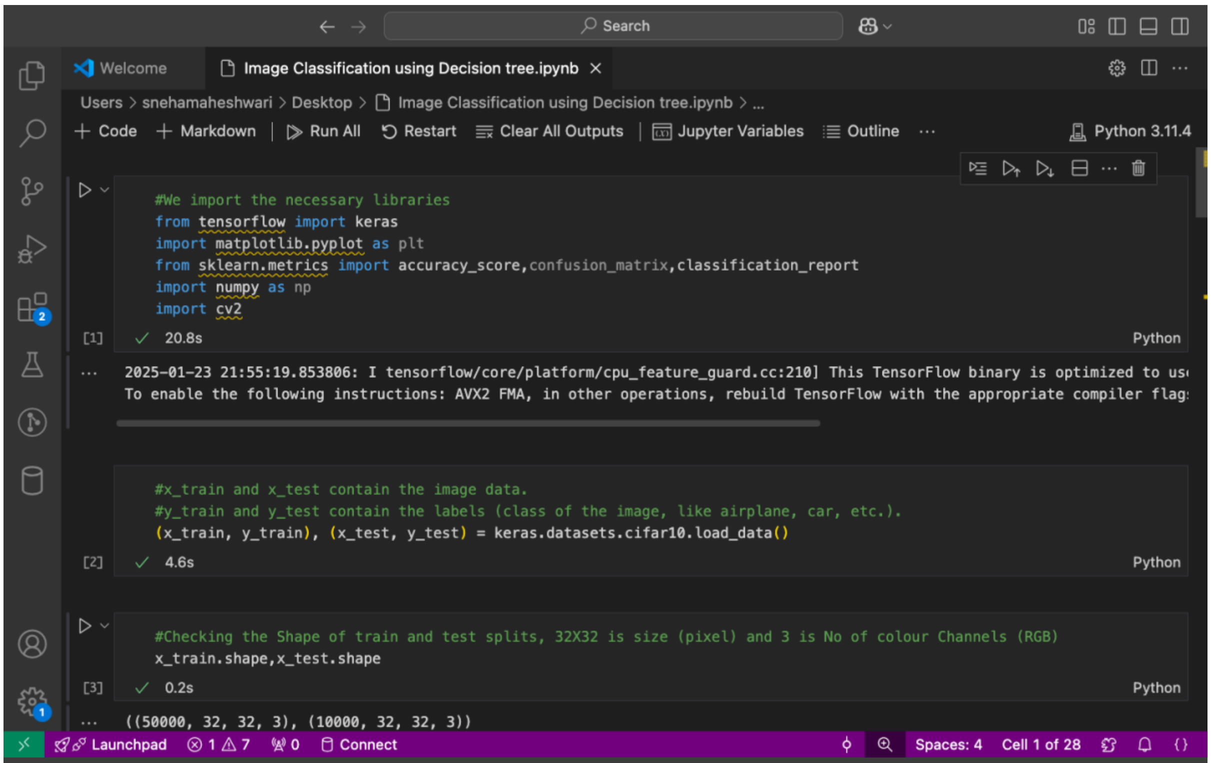
We usually use Deep Learning Algorithms such as Convoluted Neural Networks (predominantly used to extract the feature from the grid-like matrix dataset. For example, visual datasets like images or videos where data patterns play an extensive role.) or Support Vector Machines (They distinguish between two classes by finding the optimal hyperplane that maximizes the margin between the closest data points of opposite classes.), if the problem is still to be tackled using ML Algorithms.

For the case of the Decision tree, we cannot treat it as a regression problem because:

1. the predicted output here is a continuous value which is not the case for the different classes in Image Classification.
2. Decision trees are prone to **overfitting** when faced with high-dimensional data, especially when the regression target is continuous.
3. The model might treat the numeric values as ordinal or continuous. This can lead to: Predicting values "between classes" that have no real-world meaning (e.g., predicting 1.5 when there’s no such class between "dog" and "bird").

4.DecisionTreeClassifier uses a splitting criterion like **Gini Impurity** or **Entropy (Information Gain)** that is specifically designed for classification tasks. DecisionTreeRegressor's splitting criteria (like MSE) don't separate classes as effectively, leading to worse performance in classification problems.

Decision Tree for Image Classification:


Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Python 3.11.4
Launchpad ⊗ 1 ⚠ 7 0 Connect
Spaces: 4 Cell 1 of 28

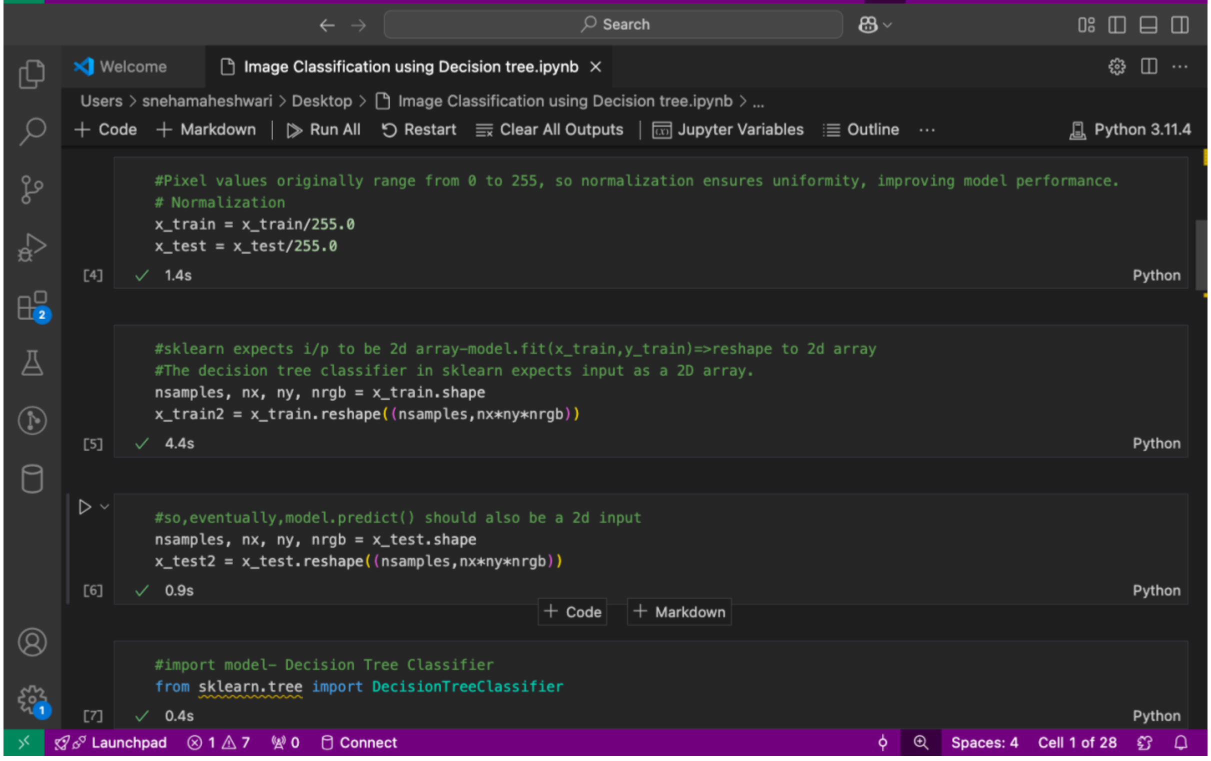
```
#We import the necessary libraries
from tensorflow import keras
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
import numpy as np
import cv2
```
[1] ✓ 20.8s Python

```
2025-01-23 21:55:19.853806: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to us
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flag
```

```
#x_train and x_test contain the image data.
#y_train and y_test contain the labels (class of the image, like airplane, car, etc.).
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
```
[2] ✓ 4.6s Python

```
#Checking the Shape of train and test splits, 32X32 is size (pixel) and 3 is No of colour Channels (RGB)
x_train.shape,x_test.shape
```
[3] ✓ 0.2s Python

```
((50000, 32, 32, 3), (10000, 32, 32, 3))
```

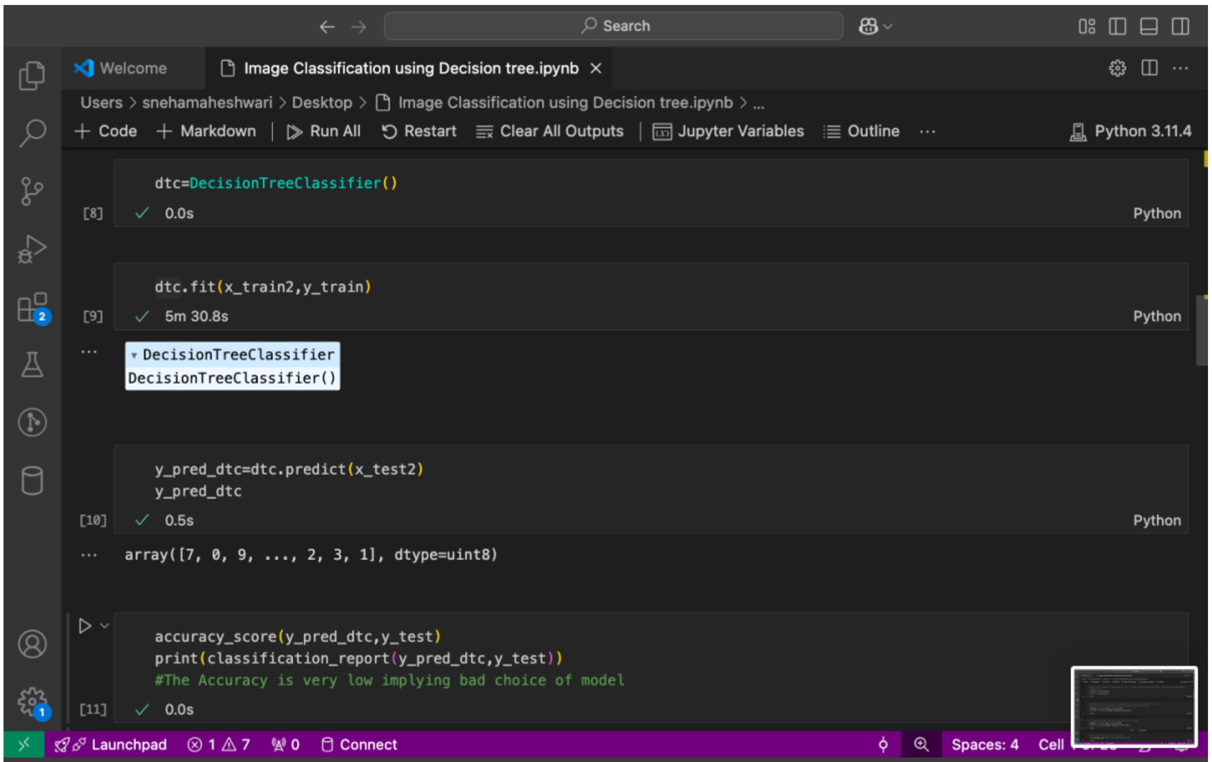

Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Python 3.11.4
+ Code + Markdown
Launchpad ⊗ 1 ⚠ 7 0 Connect
Spaces: 4 Cell 1 of 28

```
#Pixel values originally range from 0 to 255, so normalization ensures uniformity, improving model performance.
# Normalization
x_train = x_train/255.0
x_test = x_test/255.0
```
[4] ✓ 1.4s Python

```
#sklearn expects i/p to be 2d array-model.fit(x_train,y_train)=>reshape to 2d array
#The decision tree classifier in sklearn expects input as a 2D array.
nsamples, nx, ny, nrgb = x_train.shape
x_train2 = x_train.reshape((nsamples,nx*ny*nrgb))
```
[5] ✓ 4.4s Python

```
#so,eventually,model.predict() should also be a 2d input
nsamples, nx, ny, nrgb = x_test.shape
x_test2 = x_test.reshape((nsamples,nx*ny*nrgb))
```
[6] ✓ 0.9s Python

```
#import model- Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
```
[7] ✓ 0.4s Python


Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Python 3.11.4
Launchpad 1 7 0 Connect
Spaces: 4 Cell

```
dtc=DecisionTreeClassifier()
```
[8] ✓ 0.0s Python

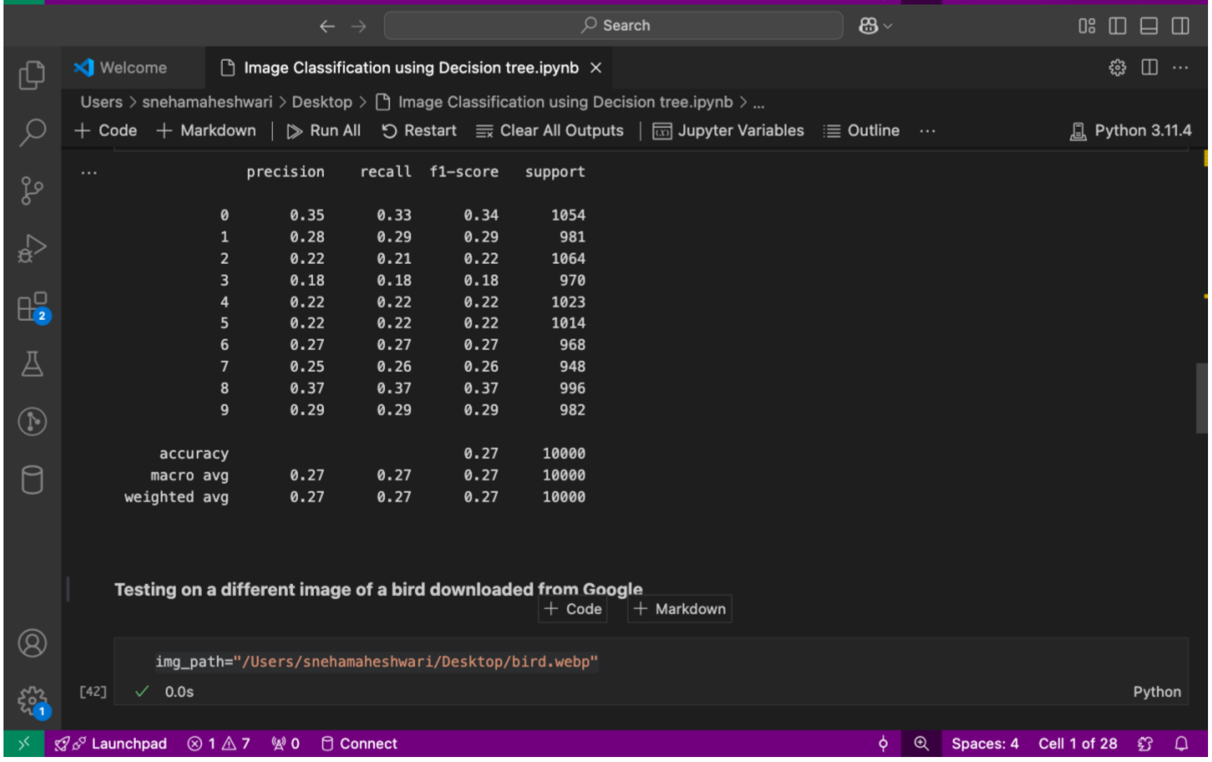
```
dtc.fit(x_train2,y_train)
```
[9] ✓ 5m 30.8s Python

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred_dtc=dtc.predict(x_test2)
y_pred_dtc
```
[10] ✓ 0.5s Python

```
array([7, 0, 9, ..., 2, 3, 1], dtype=uint8)
```

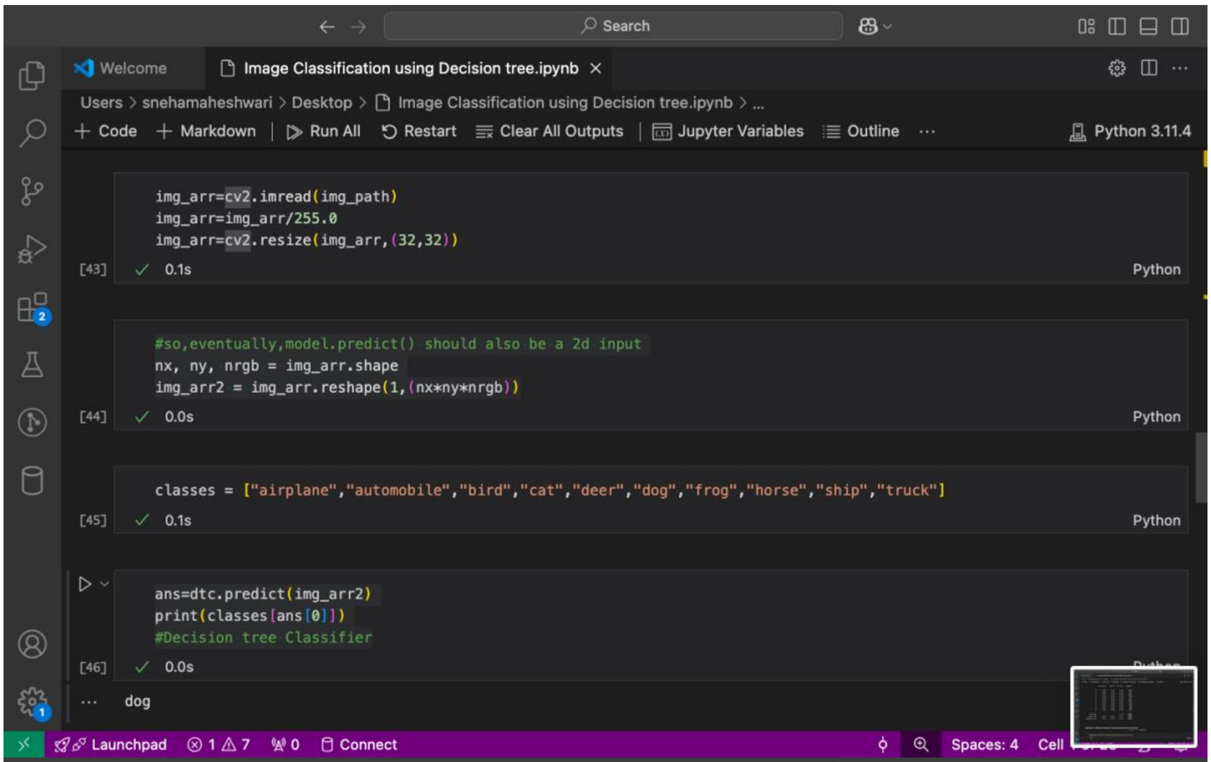
```
accuracy_score(y_pred_dtc,y_test)
print(classification_report(y_pred_dtc,y_test))
#The Accuracy is very low implying bad choice of model
```
[11] ✓ 0.0s


Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Python 3.11.4
Launchpad 1 7 0 Connect
Spaces: 4 Cell 1 of 28

```
              precision    recall  f1-score   support

           0       0.35      0.33      0.34      1054
           1       0.28      0.29      0.29       981
           2       0.22      0.21      0.22      1064
           3       0.18      0.18      0.18       970
           4       0.22      0.22      0.22      1023
           5       0.22      0.22      0.22      1014
           6       0.27      0.27      0.27       968
           7       0.25      0.26      0.26       948
           8       0.37      0.37      0.37       996
           9       0.29      0.29      0.29       982

    accuracy                           0.27     10000
   macro avg       0.27      0.27      0.27     10000
weighted avg       0.27      0.27      0.27     10000
```

Testing on a different image of a bird downloaded from Google

```
img_path="/Users/snehamaheshwari/Desktop/bird.webp"
```
[42] ✓ 0.0s Python



Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Python 3.11.4

```
img_arr=cv2.imread(img_path)
img_arr=img_arr/255.0
img_arr=cv2.resize(img_arr,(32,32))
```
[43] ✓ 0.1s Python

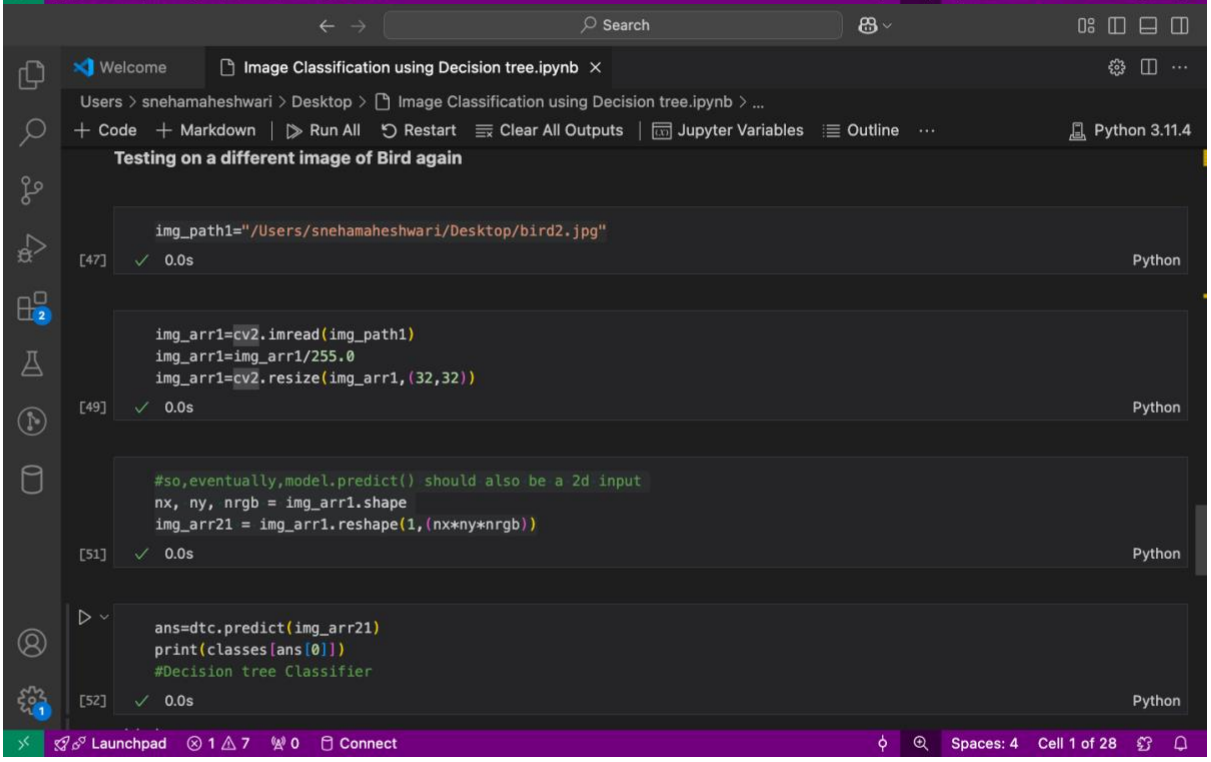
```
#so,eventually,model.predict() should also be a 2d input
nx, ny, nrgb = img_arr.shape
img_arr2 = img_arr.reshape(1,(nx*ny*nrgb))
```
[44] ✓ 0.0s Python

```
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```
[45] ✓ 0.1s Python

```
ans=dtc.predict(img_arr2)
print(classes[ans[0]])
#Decision tree Classifier
```
[46] ✓ 0.0s

```
dog
```

Launchpad ⊗ 1 ⚠ 7 0 Connect
Spaces: 4 Cell



Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
Python 3.11.4

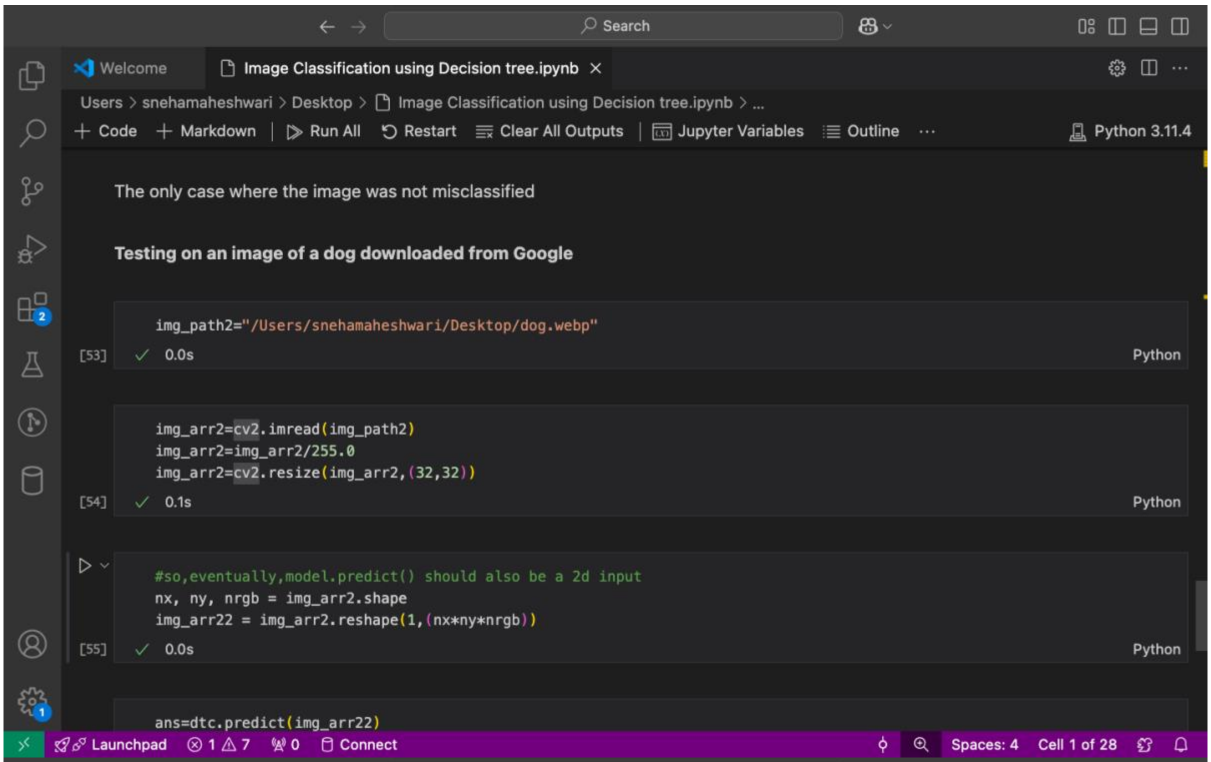
Testing on a different image of Bird again

```
img_path1="/Users/snehamaheshwari/Desktop/bird2.jpg"
```
[47] ✓ 0.0s Python

```
img_arr1=cv2.imread(img_path1)
img_arr1=img_arr1/255.0
img_arr1=cv2.resize(img_arr1,(32,32))
```
[49] ✓ 0.0s Python

```
#so,eventually,model.predict() should also be a 2d input
nx, ny, nrgb = img_arr1.shape
img_arr21 = img_arr1.reshape(1,(nx*ny*nrgb))
```
[51] ✓ 0.0s Python

```
ans=dtc.predict(img_arr21)
print(classes[ans[0]])
#Decision tree Classifier
```
[52] ✓ 0.0s Python



Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3.11.4

The only case where the image was not misclassified

Testing on an image of a dog downloaded from Google

```
img_path2="/Users/snehamaheshwari/Desktop/dog.webp"
```

[53] ✓ 0.0s Python

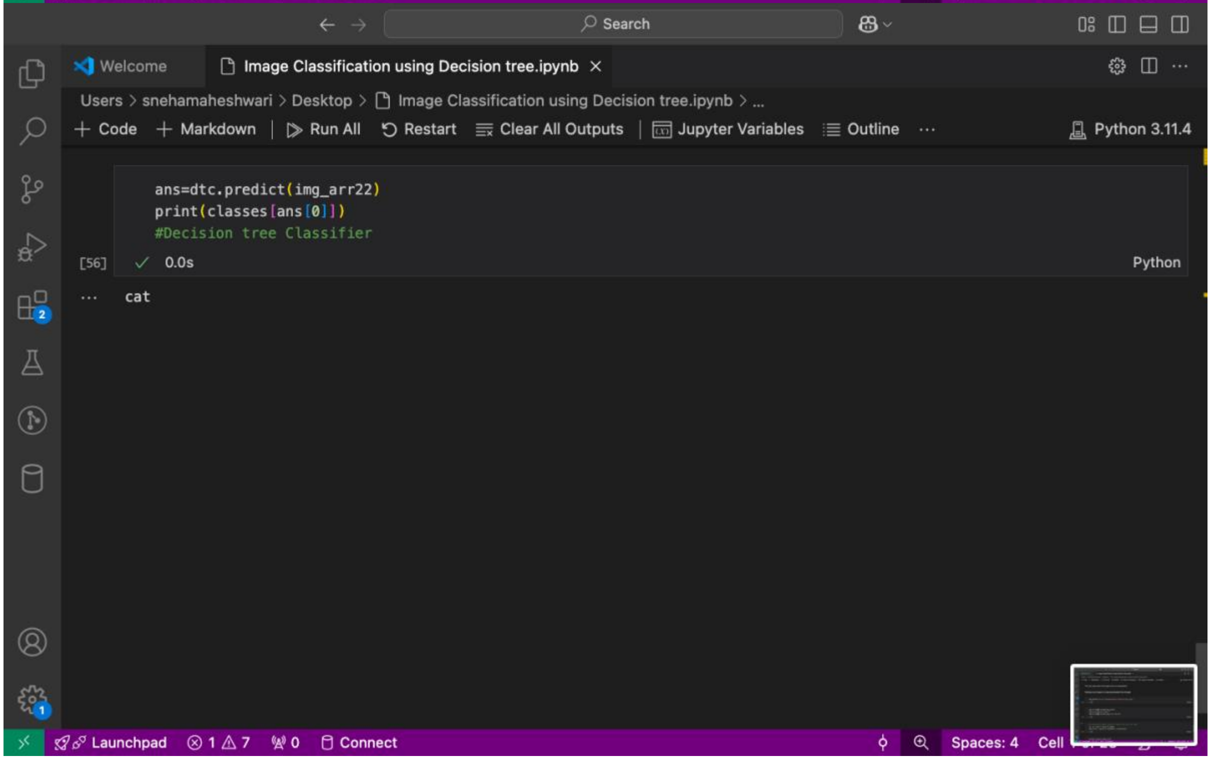
```
img_arr2=cv2.imread(img_path2)
img_arr2=img_arr2/255.0
img_arr2=cv2.resize(img_arr2,(32,32))
```

[54] ✓ 0.1s Python

```
#so,eventually,model.predict() should also be a 2d input
nx, ny, nrgb = img_arr2.shape
img_arr22 = img_arr2.reshape(1,(nx*ny*nrgb))
```

[55] ✓ 0.0s Python

```
ans=dtc.predict(img_arr22)
```



Search
Welcome
Image Classification using Decision tree.ipynb
Users > snehamaheshwari > Desktop > Image Classification using Decision tree.ipynb > ...
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3.11.4

```
ans=dtc.predict(img_arr22)
print(classes[ans[0]])
#Decision tree Classifier
```

[56] ✓ 0.0s Python

```
cat
```

RECOMMENDATION SYSTEM USING DECISION TREE

A major weakness in using decision trees as a prediction model in RS is the need to build a huge number of trees (either for each item or for each user). Moreover, the model can only compute the expected rating of a single item at a time. To provide recommendations to the user, we must traverse the tree(s) from root to leaf once for each item in order to compute its predicted rating. Only after computing the predicted rating of all items can the RS provide the recommendations (highest predicted rating items). Thus decision trees in RS do not scale well with respect to the number of items. Source.

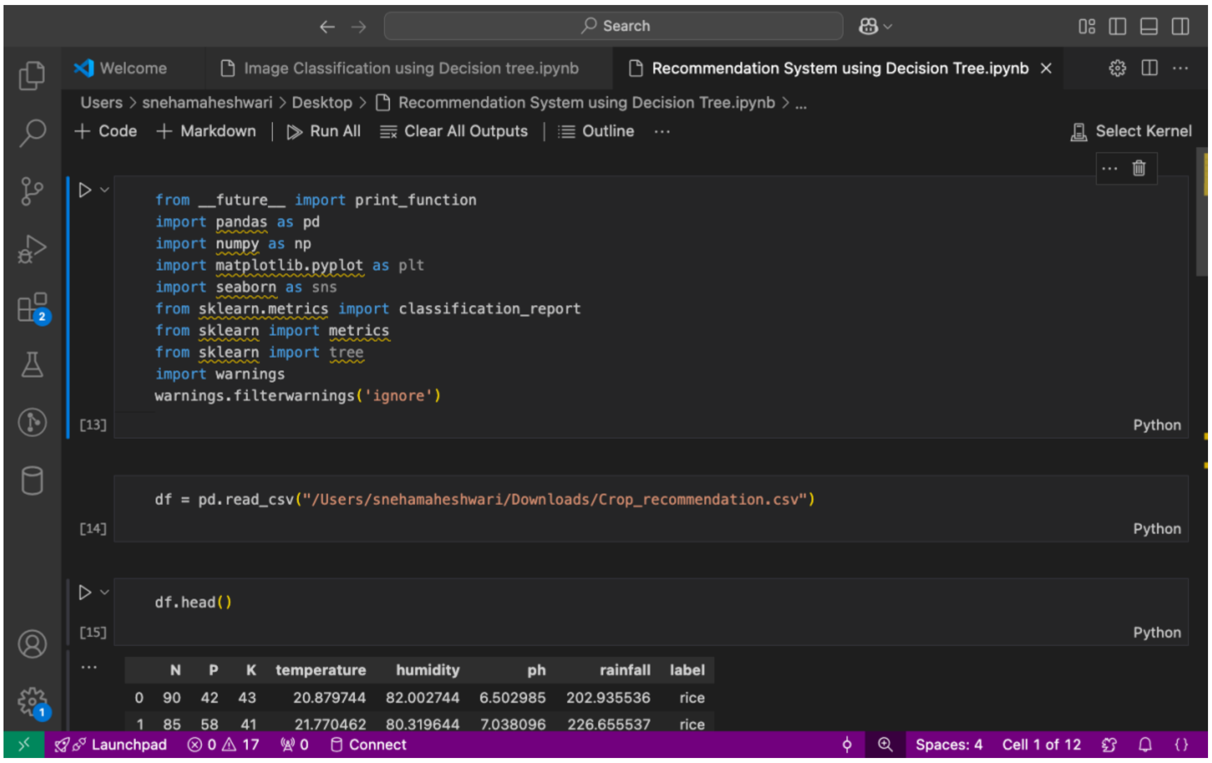
Imagine you want to recommend movies to a user, and there are 10,000 movies.
For decision trees:

1. You have to calculate a score (rating) for each movie one by one.
2. This means **going through the tree** separately for each movie, which takes a lot of time.

Now imagine you have 1,000 users and 10,000 movies. You'd have to do this process **1,000 × 10,000 = 10 million calculations** just to provide recommendations.

Let us consider an example of building a Crop Recommendation system based on:

- N - ratio of Nitrogen content in soil
- P - ratio of Phosphorous content in soil
- K - ratio of Potassium content in soil
- temperature - temperature in degree Celsius
- humidity - relative humidity in %
- ph - ph value of the soil
- rainfall - rainfall in mm



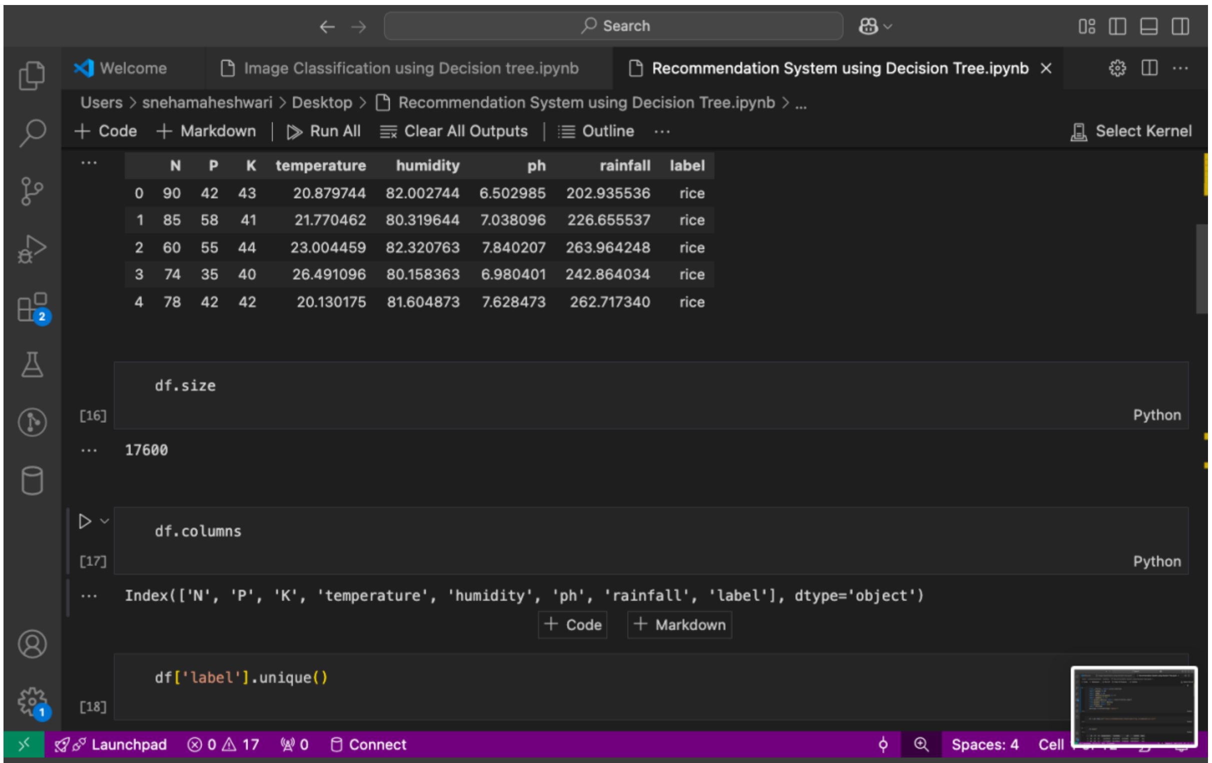
Search
Welcome
Image Classification using Decision tree.ipynb
Recommendation System using Decision Tree.ipynb
Users > snehamaheshwari > Desktop > Recommendation System using Decision Tree.ipynb > ...
+ Code
+ Markdown
Run All
Clear All Outputs
Outline
Select Kernel
Python
Launchpad
Connect
Spaces: 4
Cell 1 of 12

```
[13]
from __future__ import print_function
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')
```

```
[14]
df = pd.read_csv("/Users/snehamaheshwari/Downloads/Crop_recommendation.csv")
```

```
[15]
df.head()
```

```
    N   P   K  temperature   humidity        ph    rainfall  label
0  90  42  43    20.879744  82.002744  6.502985  202.935536   rice
1  85  58  41    21.770462  80.319644  7.038096  226.655537   rice
```



Recommendation System using Decision Tree.ipynb

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

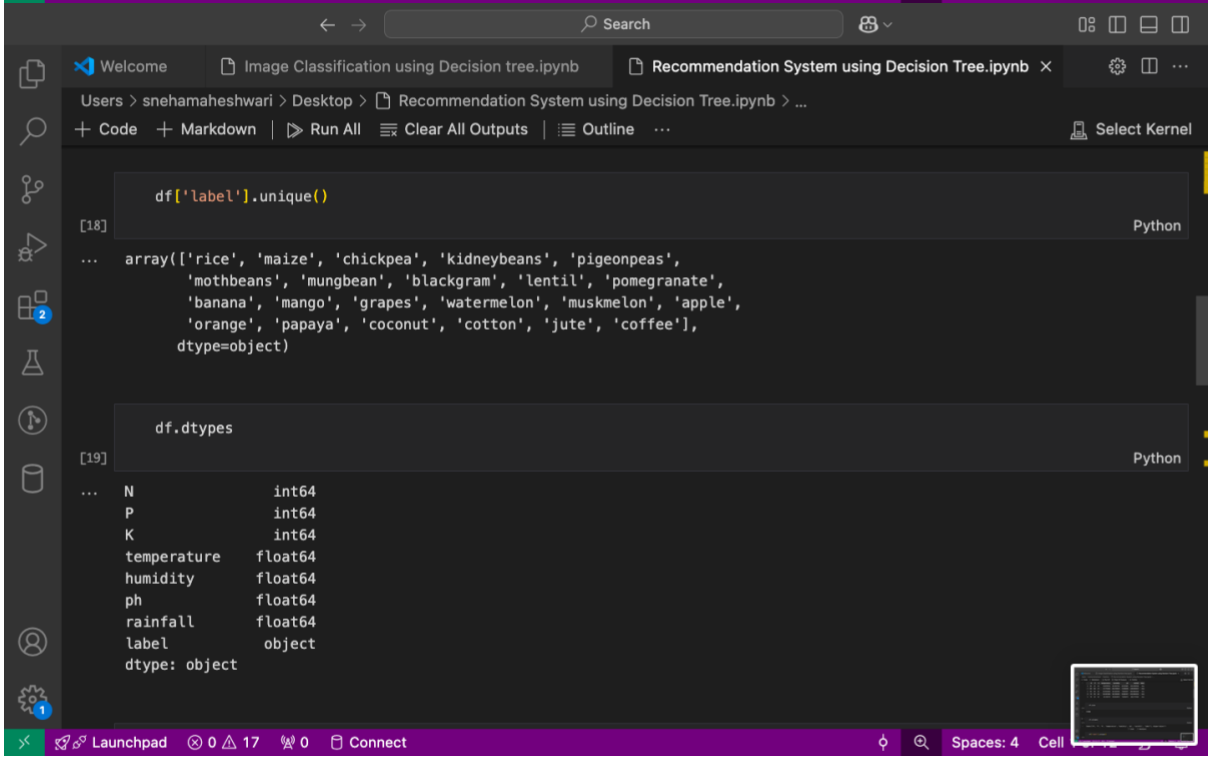
```
[16] df.size
```

```
17600
```

```
[17] df.columns
```

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

```
[18] df['label'].unique()
```

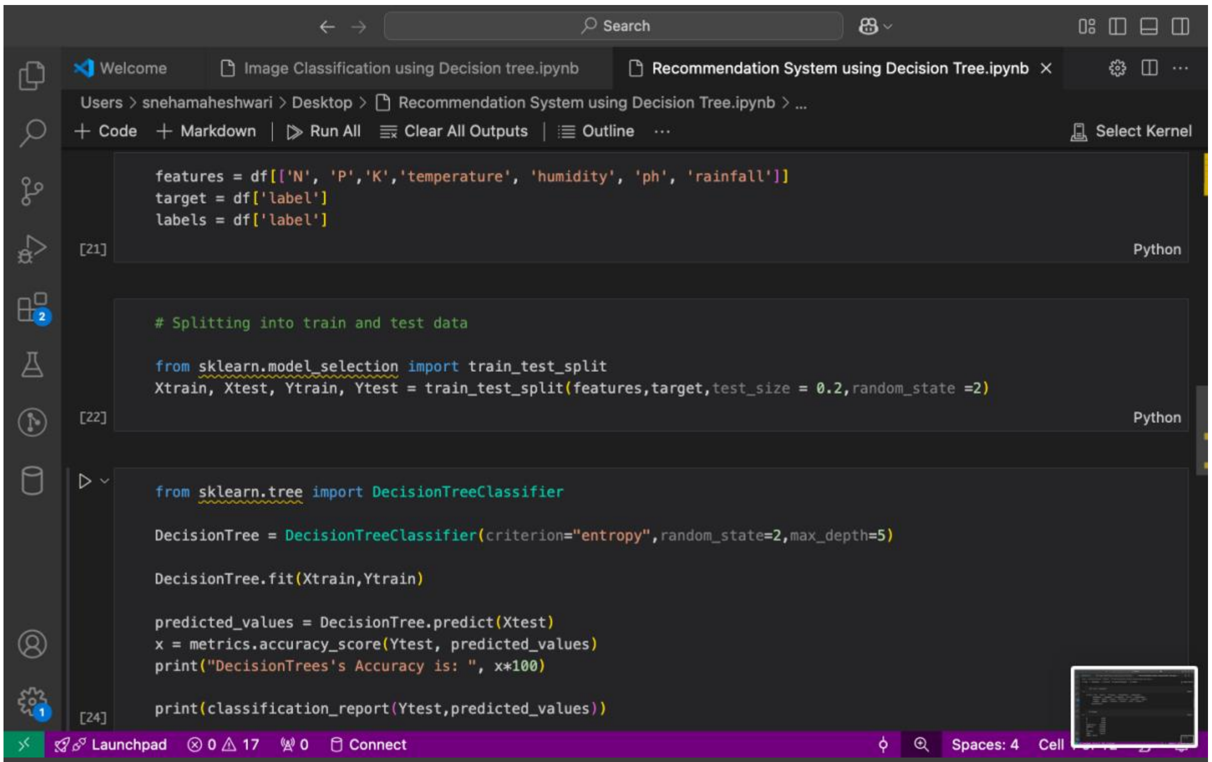


```
[18] df['label'].unique()
```

```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

```
[19] df.dtypes
```

```
N                int64
P                int64
K                int64
temperature    float64
humidity       float64
ph             float64
rainfall       float64
label           object
dtype: object
```

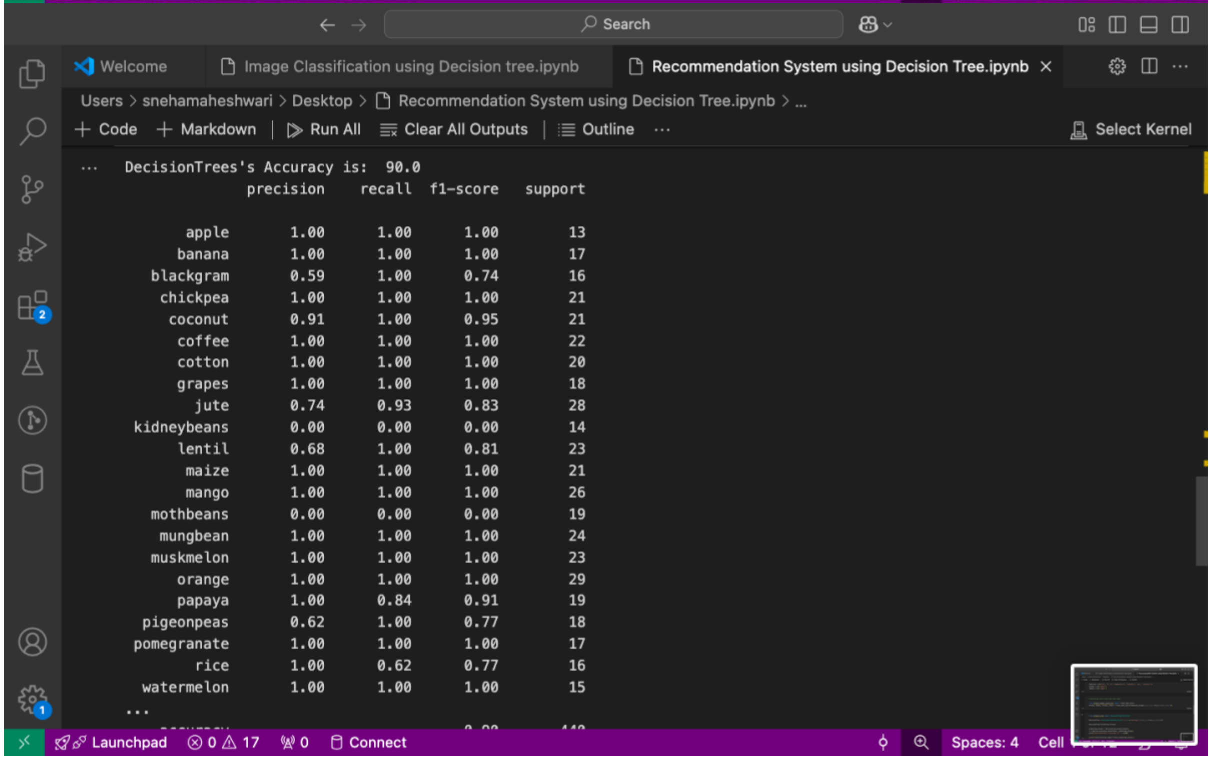


Search
Welcome
Image Classification using Decision tree.ipynb
Recommendation System using Decision Tree.ipynb
Users > snehamaheshwari > Desktop > Recommendation System using Decision Tree.ipynb > ...
Code
Markdown
Run All
Clear All Outputs
Outline
Select Kernel
Python
Launchpad
Connect
Spaces: 4
Cell

```
features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = df['label']
labels = df['label']
```
[21]

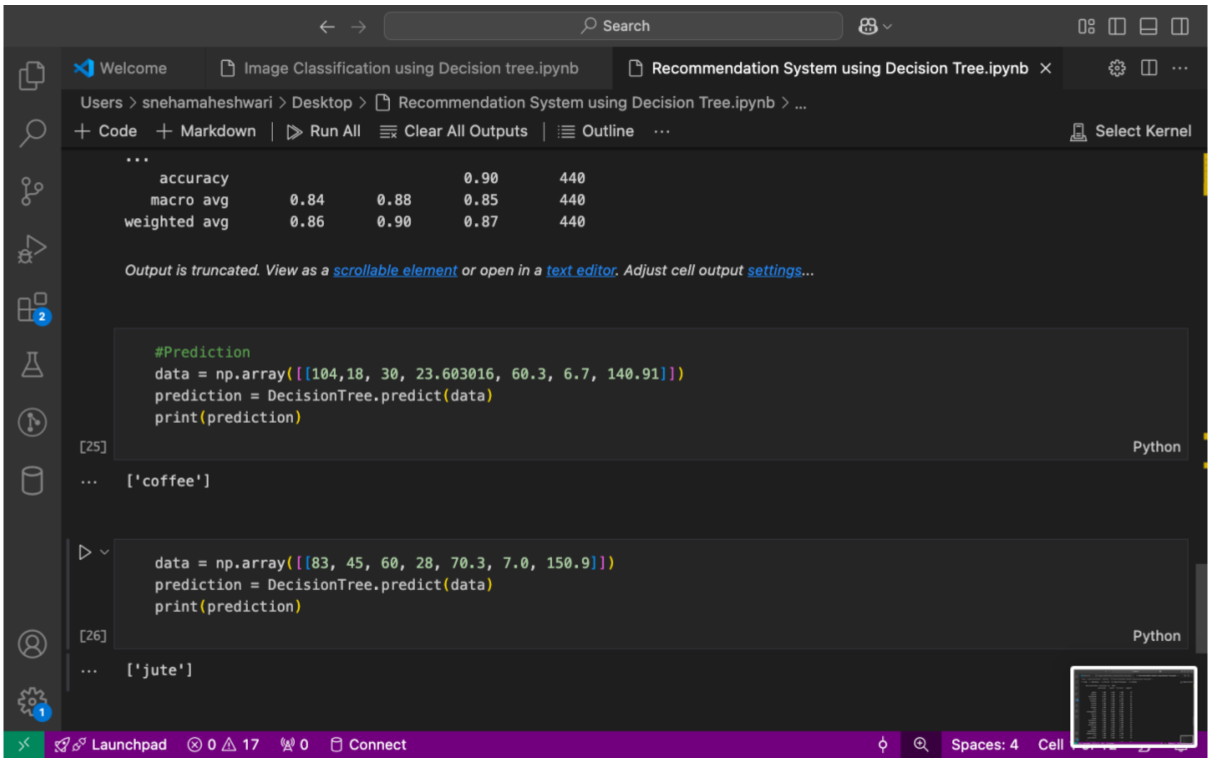
```
# Splitting into train and test data

from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,target,test_size = 0.2,random_state =2)
```
[22]

```
from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)

DecisionTree.fit(Xtrain,Ytrain)

predicted_values = DecisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
print("DecisionTrees's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))
```
[24]



Search
Welcome
Image Classification using Decision tree.ipynb
Recommendation System using Decision Tree.ipynb
Users > snehamaheshwari > Desktop > Recommendation System using Decision Tree.ipynb > ...
Code
Markdown
Run All
Clear All Outputs
Outline
Select Kernel
Launchpad
Connect
Spaces: 4
Cell

```
...  DecisionTrees's Accuracy is:  90.0
                   precision    recall  f1-score   support

           apple       1.00      1.00      1.00        13
          banana       1.00      1.00      1.00        17
       blackgram       0.59      1.00      0.74        16
        chickpea       1.00      1.00      1.00        21
         coconut       0.91      1.00      0.95        21
          coffee       1.00      1.00      1.00        22
          cotton       1.00      1.00      1.00        20
          grapes       1.00      1.00      1.00        18
            jute       0.74      0.93      0.83        28
     kidneybeans       0.00      0.00      0.00        14
          lentil       0.68      1.00      0.81        23
           maize       1.00      1.00      1.00        21
           mango       1.00      1.00      1.00        26
       mothbeans       0.00      0.00      0.00        19
        mungbean       1.00      1.00      1.00        24
       muskmelon       1.00      1.00      1.00        23
          orange       1.00      1.00      1.00        29
          papaya       1.00      0.84      0.91        19
      pigeonpeas       0.62      1.00      0.77        18
     pomegranate       1.00      1.00      1.00        17
            rice       1.00      0.62      0.77        16
      watermelon       1.00      1.00      1.00        15
 ...
```



```
...
    accuracy                        0.90       440
   macro avg       0.84      0.88   0.85       440
weighted avg       0.86      0.90   0.87       440
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
#Prediction
data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = DecisionTree.predict(data)
print(prediction)
```

```
['coffee']
```

```python
data = np.array([[83, 45, 60, 28, 70.3, 7.0, 150.9]])
prediction = DecisionTree.predict(data)
print(prediction)
```

```
['jute']
```

We are getting a good accuracy here because the dataset is not high dimensional and so the decision tree would not be very prone to overfitting.

(read: articles running decision trees on large data sets)

1. https://www.codeproject.com/Articles/5366638/Decision-Tree-Credit-Card-Fraud-Detection
2. https://www.itm-conferences.org/articles/itmconf/pdf/2023/03/itmconf_icdsia2023_02012.pdf