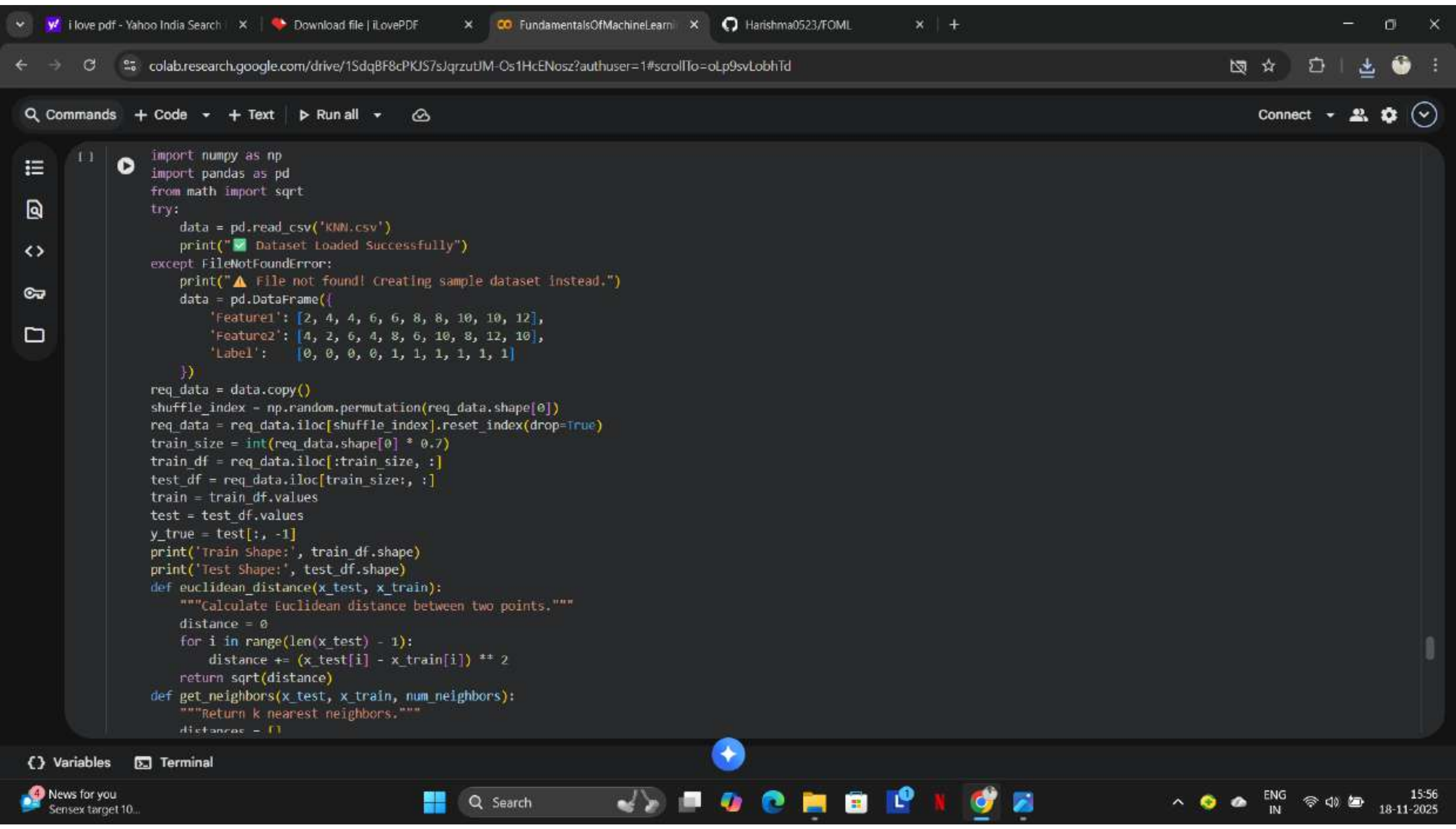
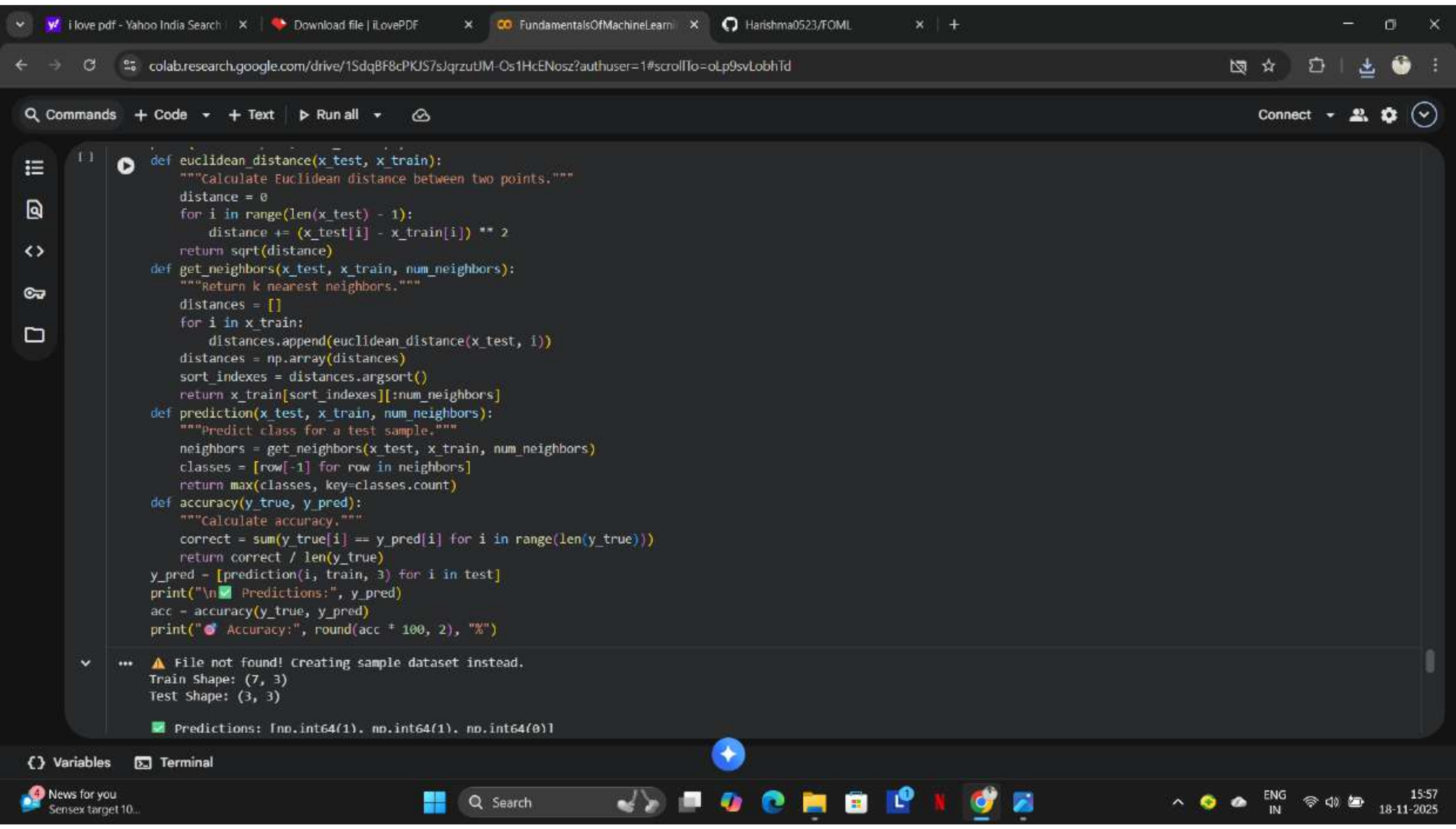

i love pdf - Yahoo India Search
Download file | iLovePDF
FundamentalsOfMachineLearni
Harishma0523/FOML
colab.research.google.com/drive/1SdqBF8cPKJS7sJqrzutJM-Os1HcENosz?authuser=1#scrollTo=oLp9svLobhTd
Commands
+ Code
+ Text
Run all
Connect
Variables
Terminal
News for you
Sensex target 10...
Search
ENG
IN
15:56
18-11-2025

```python
import numpy as np
import pandas as pd
from math import sqrt
try:
    data = pd.read_csv('KNN.csv')
    print("✅ Dataset Loaded Successfully")
except FileNotFoundError:
    print("⚠ File not found! Creating sample dataset instead.")
    data = pd.DataFrame({
        'Feature1': [2, 4, 4, 6, 6, 8, 8, 10, 10, 12],
        'Feature2': [4, 2, 6, 4, 8, 6, 10, 8, 12, 10],
        'Label':    [0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
    })
req_data = data.copy()
shuffle_index = np.random.permutation(req_data.shape[0])
req_data = req_data.iloc[shuffle_index].reset_index(drop=True)
train_size = int(req_data.shape[0] * 0.7)
train_df = req_data.iloc[:train_size, :]
test_df = req_data.iloc[train_size:, :]
train = train_df.values
test = test_df.values
y_true = test[:, -1]
print('Train Shape:', train_df.shape)
print('Test Shape:', test_df.shape)
def euclidean_distance(x_test, x_train):
    """Calculate Euclidean distance between two points."""
    distance = 0
    for i in range(len(x_test) - 1):
        distance += (x_test[i] - x_train[i]) ** 2
    return sqrt(distance)
def get_neighbors(x_test, x_train, num_neighbors):
    """Return k nearest neighbors."""
    distances = []
```


i love pdf - Yahoo India Search
Download file | iLovePDF
FundamentalsOfMachineLearni
Harishma0523/FOML
colab.research.google.com/drive/1SdqBF8cPKJS7sJqrzutJM-Os1HcENosz?authuser=1#scrollTo=oLp9svLobhTd
Commands
+ Code
+ Text
Run all
Connect
Variables
Terminal
News for you
Sensex target 10...
Search
ENG
IN
15:57
18-11-2025

```python
def euclidean_distance(x_test, x_train):
    """Calculate Euclidean distance between two points."""
    distance = 0
    for i in range(len(x_test) - 1):
        distance += (x_test[i] - x_train[i]) ** 2
    return sqrt(distance)
def get_neighbors(x_test, x_train, num_neighbors):
    """Return k nearest neighbors."""
    distances = []
    for i in x_train:
        distances.append(euclidean_distance(x_test, i))
    distances = np.array(distances)
    sort_indexes = distances.argsort()
    return x_train[sort_indexes][:num_neighbors]
def prediction(x_test, x_train, num_neighbors):
    """Predict class for a test sample."""
    neighbors = get_neighbors(x_test, x_train, num_neighbors)
    classes = [row[-1] for row in neighbors]
    return max(classes, key=classes.count)
def accuracy(y_true, y_pred):
    """Calculate accuracy."""
    correct = sum(y_true[i] == y_pred[i] for i in range(len(y_true)))
    return correct / len(y_true)
y_pred = [prediction(i, train, 3) for i in test]
print("\n✅ Predictions:", y_pred)
acc = accuracy(y_true, y_pred)
print("🎯 Accuracy:", round(acc * 100, 2), "%")
```

```
⚠ File not found! Creating sample dataset instead.
Train Shape: (7, 3)
Test Shape: (3, 3)

✅ Predictions: [np.int64(1), np.int64(1), np.int64(0)]
```