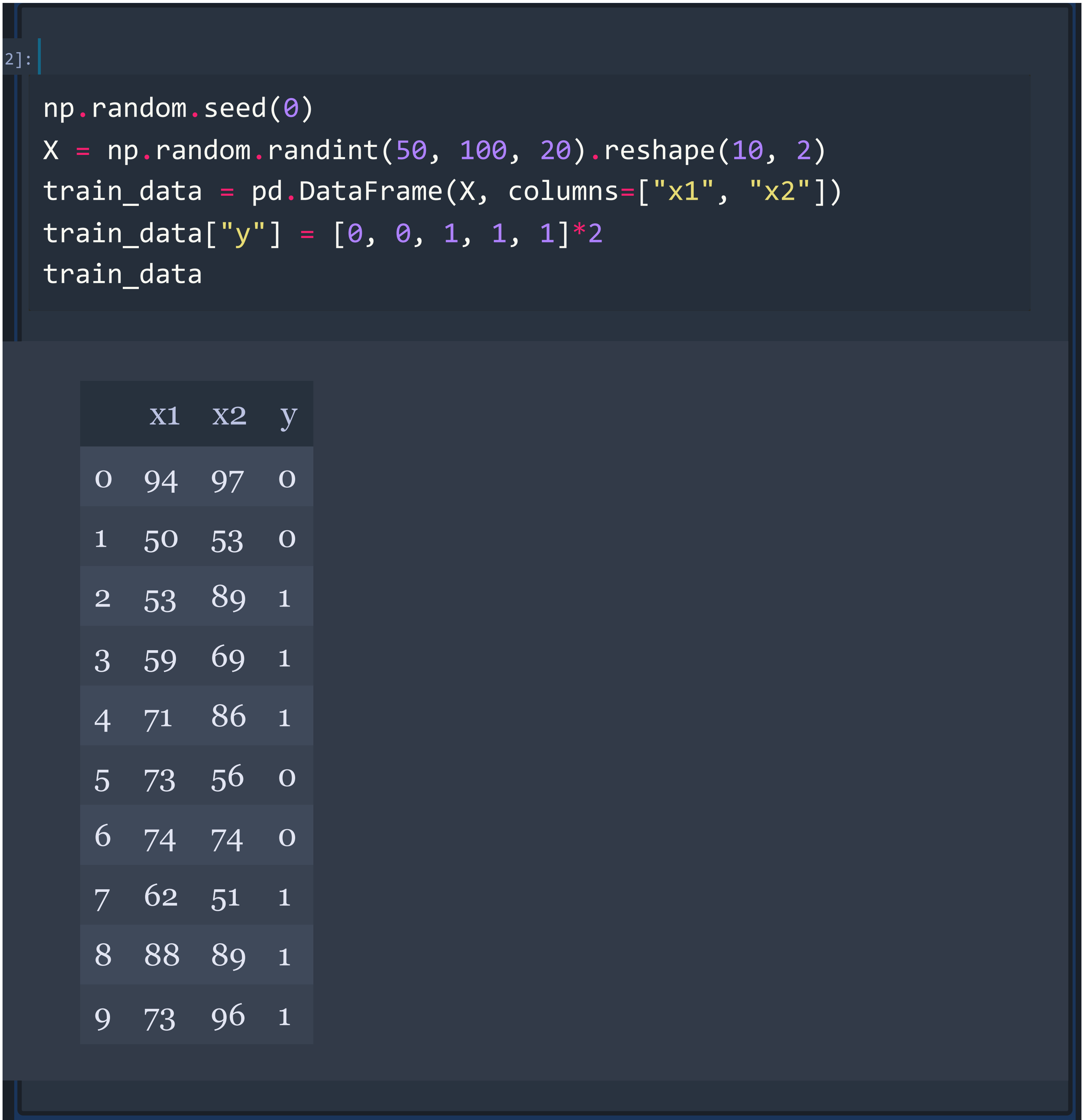
1]:

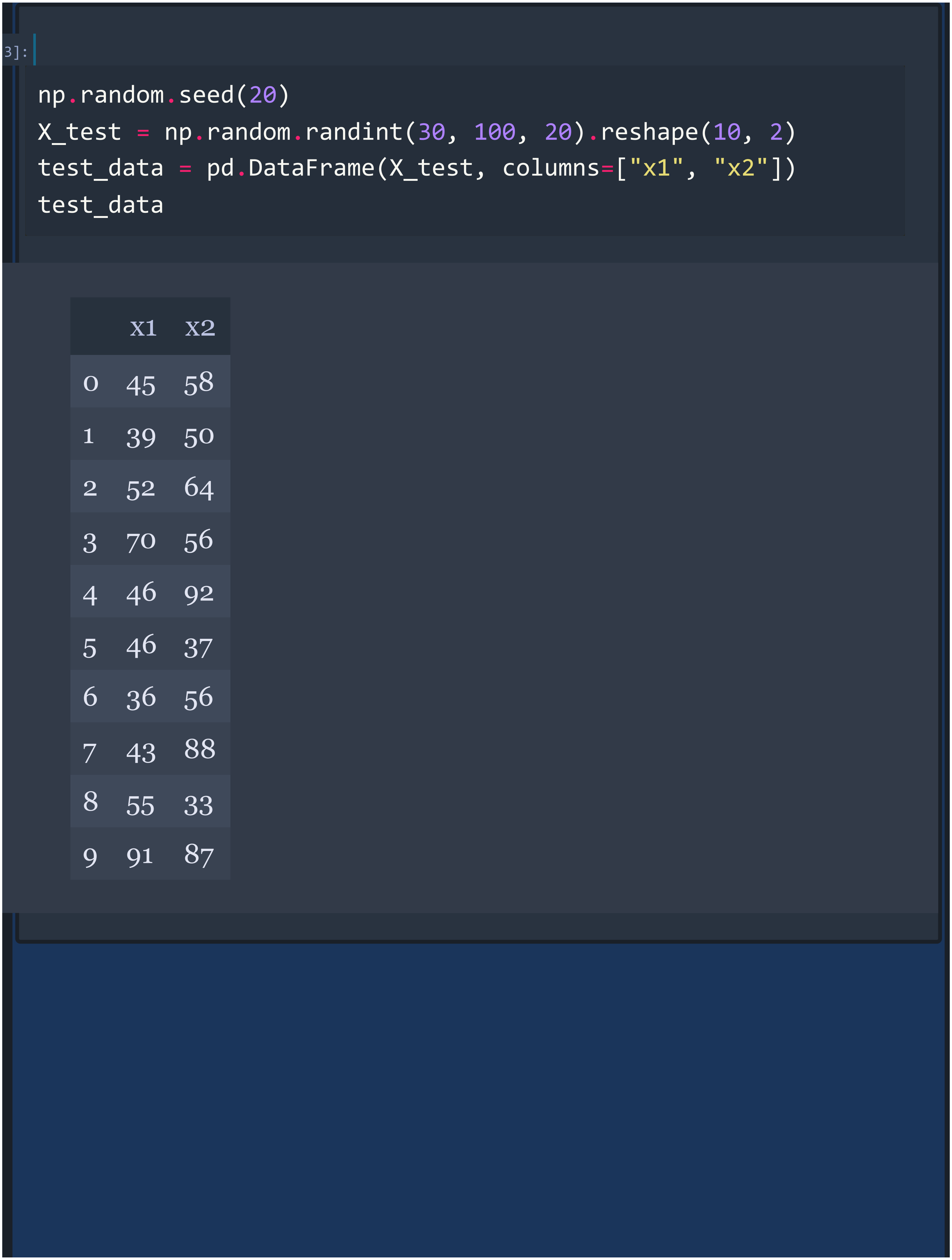
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```



2]:

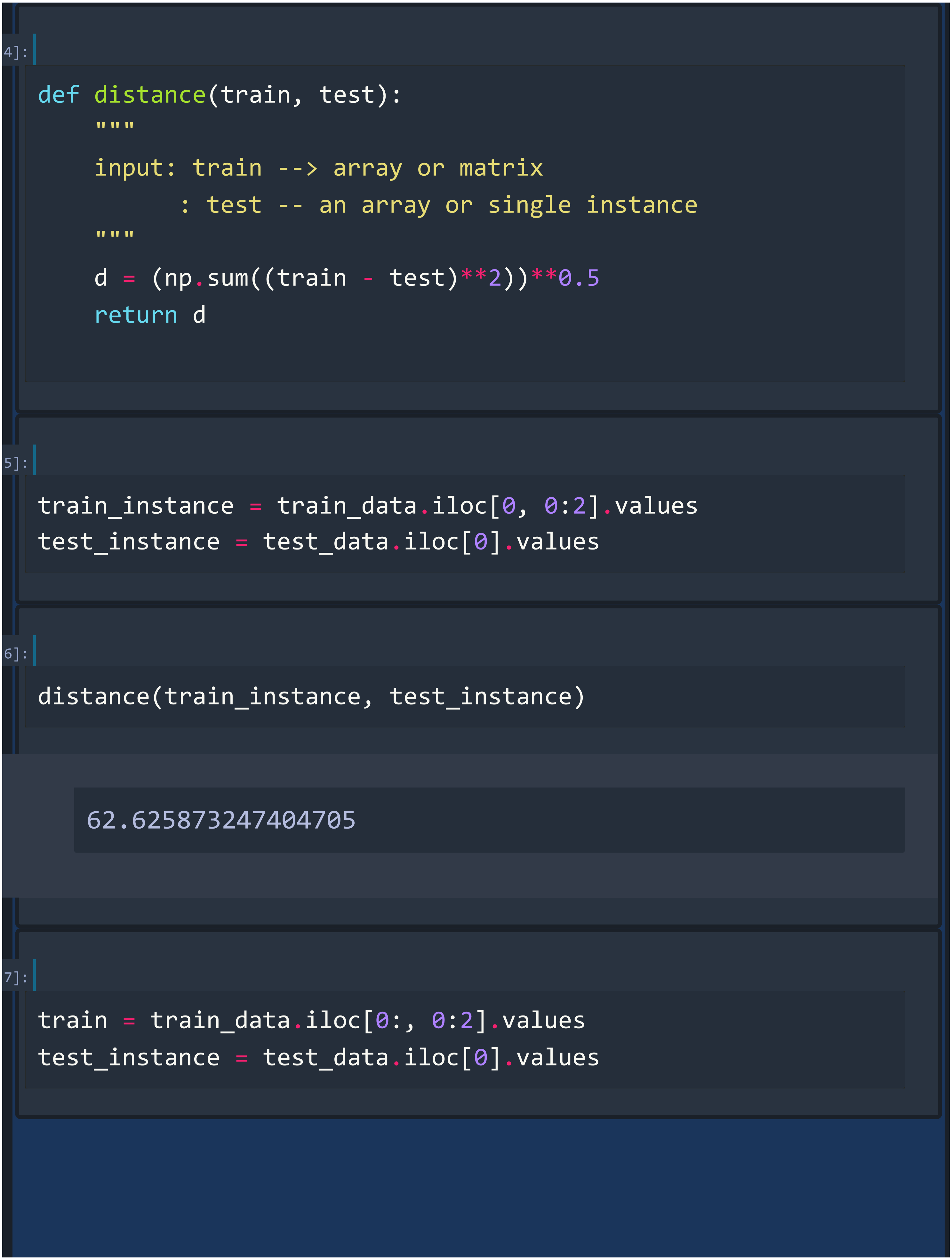
```
np.random.seed(0)
X = np.random.randint(50, 100, 20).reshape(10, 2)
train_data = pd.DataFrame(X, columns=["x1", "x2"])
train_data["y"] = [0, 0, 1, 1, 1]*2
train_data
```

	x1	x2	y
0	94	97	0
1	50	53	0
2	53	89	1
3	59	69	1
4	71	86	1
5	73	56	0
6	74	74	0
7	62	51	1
8	88	89	1
9	73	96	1



```
np.random.seed(20)
X_test = np.random.randint(30, 100, 20).reshape(10, 2)
test_data = pd.DataFrame(X_test, columns=["x1", "x2"])
test_data
```

	x1	x2
0	45	58
1	39	50
2	52	64
3	70	56
4	46	92
5	46	37
6	36	56
7	43	88
8	55	33
9	91	87



4]:

```
def distance(train, test):
    """
    input: train --> array or matrix
          : test -- an array or single instance
    """
    d = (np.sum((train - test)**2))**0.5
    return d
```

5]:

```
train_instance = train_data.iloc[0, 0:2].values
test_instance = test_data.iloc[0].values
```

6]:

```
distance(train_instance, test_instance)
```

```
62.625873247404705
```

7]:

```
train = train_data.iloc[0:, 0:2].values
test_instance = test_data.iloc[0].values
```

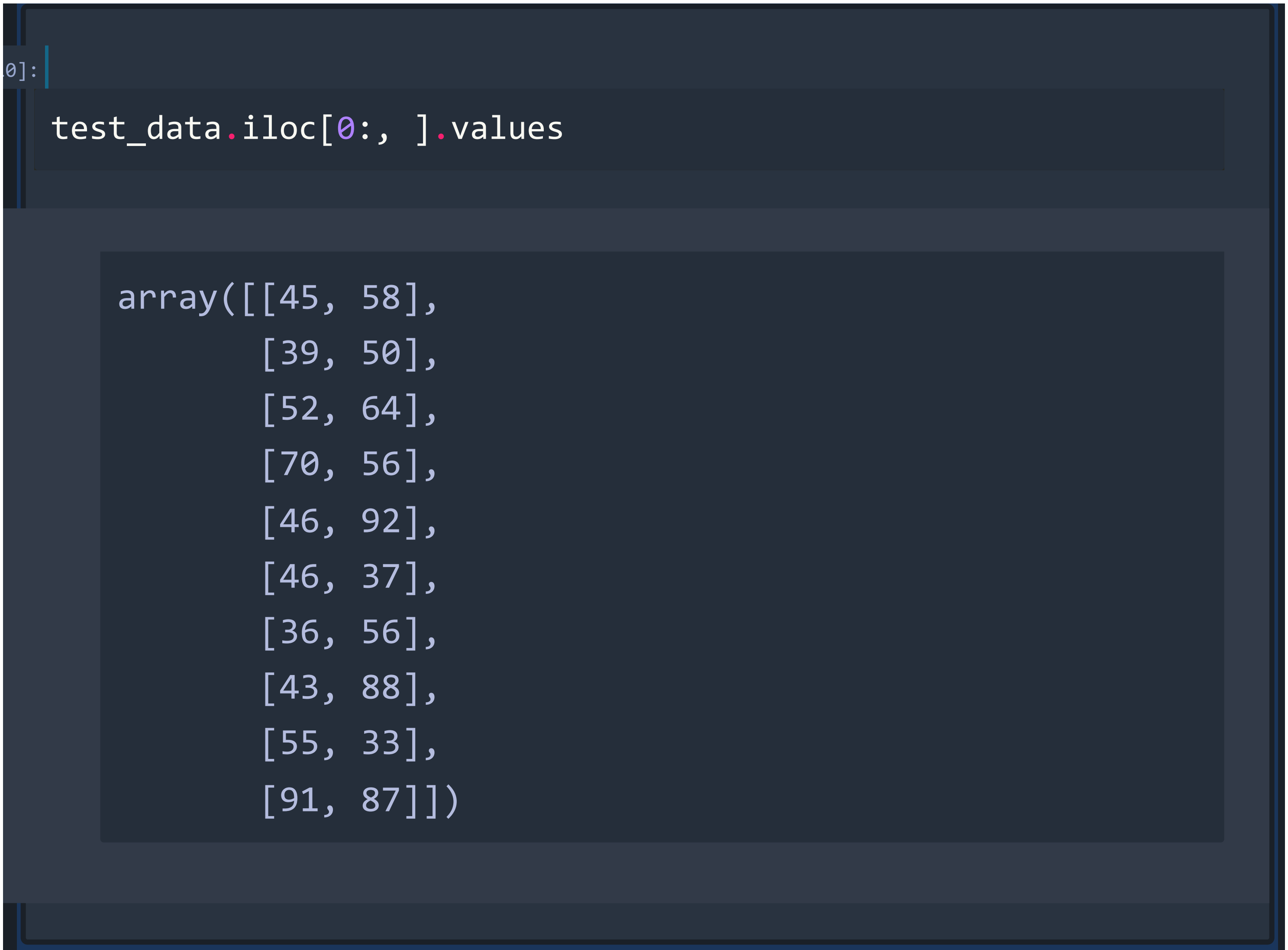
8]:

```
train_data.iloc[0:, 0:2].values
```

```
array([[94, 97],
       [50, 53],
       [53, 89],
       [59, 69],
       [71, 86],
       [73, 56],
       [74, 74],
       [62, 51],
       [88, 89],
       [73, 96]])
```

9]:

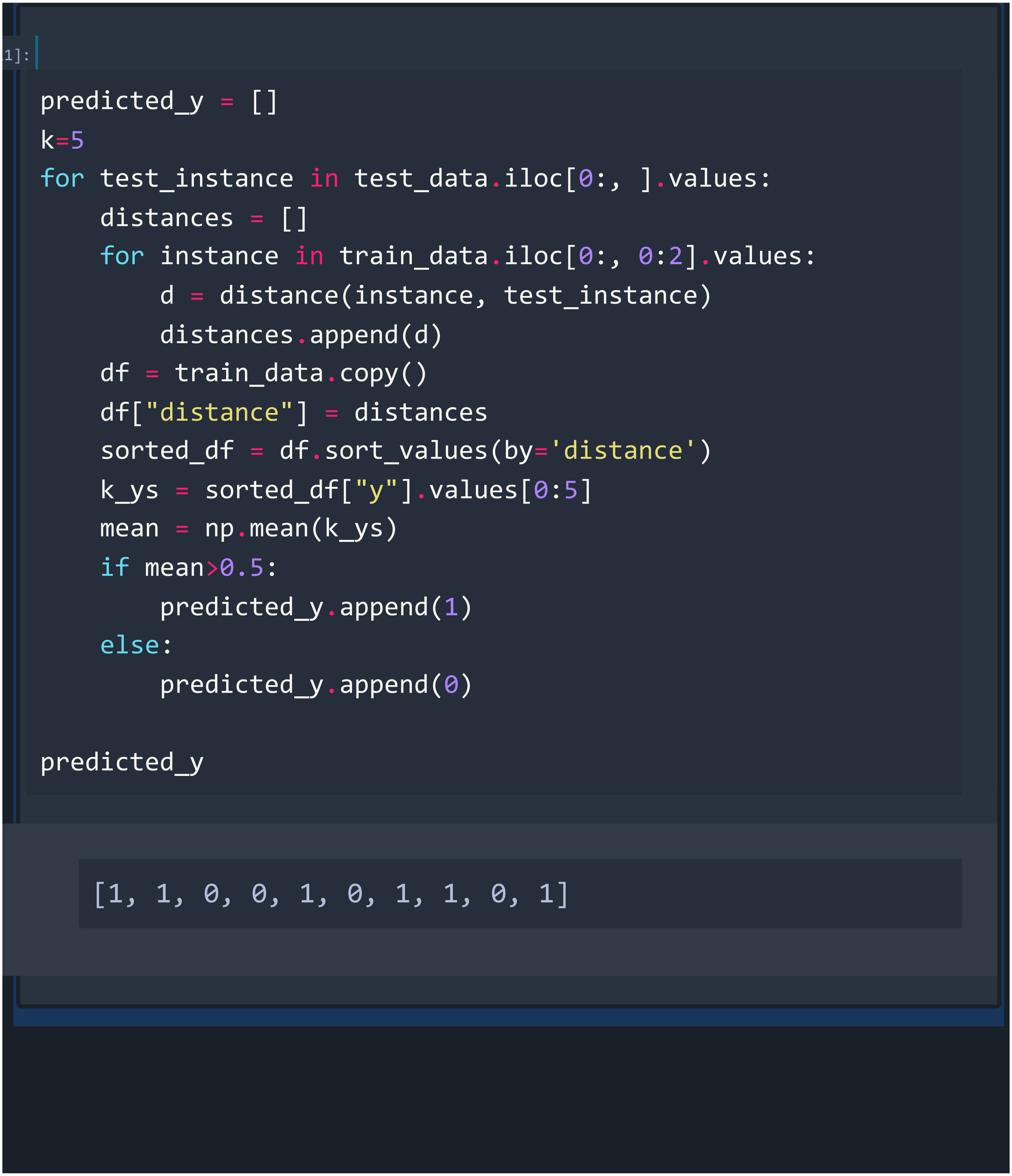
```
train = train_data.iloc[0:, 0:2].values
test_instance = test_data.iloc[0].values
```



0]:

```
test_data.iloc[0:, ].values
```

```
array([[45, 58],
       [39, 50],
       [52, 64],
       [70, 56],
       [46, 92],
       [46, 37],
       [36, 56],
       [43, 88],
       [55, 33],
       [91, 87]])
```



1]:

```
predicted_y = []
k=5
for test_instance in test_data.iloc[0:, ].values:
    distances = []
    for instance in train_data.iloc[0:, 0:2].values:
        d = distance(instance, test_instance)
        distances.append(d)
    df = train_data.copy()
    df["distance"] = distances
    sorted_df = df.sort_values(by='distance')
    k_ys = sorted_df["y"].values[0:5]
    mean = np.mean(k_ys)
    if mean>0.5:
        predicted_y.append(1)
    else:
        predicted_y.append(0)

predicted_y
```

```
[1, 1, 0, 0, 1, 0, 1, 1, 0, 1]
```