The rule is that if you're multiplying an array from the left, the array must have the same number of elements as there are rows in the matrix. And if you're multiplying the *matrix* from the left, the number of columns in the matrix must equal the number of elements in the array on the right.

Making a column vector

You see above that sometimes you'll want a column vector, even though by default NumPy arrays work like row vectors. It's possible to get the transpose of an array like

so `arr.T`, but for a 1D array, the transpose will return a row vector. Instead, use `arr[:,None]` to create a column vector:

```
print(features)
> array([ 0.49671415, -0.1382643 ,  0.64768854])

print(features.T)
> array([ 0.49671415, -0.1382643 ,  0.64768854])

print(features[:, None])
> array([[ 0.49671415],
       [-0.1382643 ],
       [ 0.64768854]])
```

Alternatively, you can create arrays with two dimensions. Then, you can use `arr.T` to get the column vector.

```
np.array(features, ndmin=2)
> array([[ 0.49671415, -0.1382643 ,  0.64768854]])

np.array(features, ndmin=2).T
> array([[ 0.49671415],
       [-0.1382643 ],
       [ 0.64768854]])
```

I personally prefer keeping all vectors as 1D arrays, it just works better in my head.