A pattern attempted to extract an incorrect number of fields from a variant.

Erroneous code example:

```
enum Fruit {
    Apple(String, String),
    Pear(u32),
}

let x = Fruit::Apple(String::new(), String::new());

match x {
    Fruit::Apple(a) => {}, // error!
    _ => {}
}
```

A pattern used to match against an enum variant must provide a sub-pattern for each field of the enum variant.

Here the `Apple` variant has two fields, and should be matched against like so:

```
enum Fruit {
    Apple(String, String),
    Pear(u32),
}

let x = Fruit::Apple(String::new(), String::new());

// Correct.
match x {
    Fruit::Apple(a, b) => {},
    _ => {}
}
```

Matching with the wrong number of fields has no sensible interpretation:

```
enum Fruit {
    Apple(String, String),
    Pear(u32),
}

let x = Fruit::Apple(String::new(), String::new());

// Incorrect.
match x {
    Fruit::Apple(a) => {},
    Fruit::Apple(a, b, c) => {},
}
```

Check how many fields the enum was declared with and ensure that your pattern uses the same number.