Ans 2

The Dummy Variable Trap occurs when two or more dummy variables created by one-hot encoding are highly correlated (multi-collinear). This means that one variable can be predicted from the others, making it difficult to interpret predicted coefficient variables in regression models. In other words, the individual effect of the dummy variables on the prediction model cannot be interpreted well because of multi co-linearity.

To overcome the Dummy variable Trap, we drop one of the columns created when the categorical variable were converted to dummy variables by one-hot encoding. This can be done because the dummy variables include redundant information.

Ans 3

A one hot encoding is a representation of categorical variables as binary vectors.

ThisThis first requires that the categorical values be mapped to integer values.

ThenThen, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1

Assume we have a sequence of labels with the values ‘red’ and ‘green’.

We can assign ‘red’ an integer value of 0 and ‘green’ the integer value of 1. As long as we always assign these numbers to these labels, this is called an integer encoding. Consistency is important so that we can invert the encoding later and get labels back from integer values, such as in the case of making a prediction.

Next, we can create a binary vector to represent each integer value. The vector will have a length of 2 for the 2 possible integer values.

The ‘red’ label encoded as a 0 will be represented with a binary vector [1, 0] where the zeroth index is marked with a value of 1. In turn, the ‘green’ label encoded as a 1 will be represented with a binary vector [0, 1] where the first index is marked with a value of 1.

Ans 4

Noteworthy differences from C/C++

Julia arrays are indexed with square brackets, and can have more than one dimension A[i,j]. This syntax is not just syntactic sugar for a reference to a pointer or address as in C/C++. See the manual entry about array construction.

In Julia, indexing of arrays, strings, etc. is 1-based not 0-based.

Julia arrays are not copied when assigned to another variable. After A = B, changing elements of B will modify A as well. Updating operators like += do not operate in-place, they are equivalent to A = A + B which rebinds the left-hand side to the result of the right-hand side expression.

Julia arrays are column major (Fortran ordered) whereas C/C++ arrays are row major ordered by default. To get optimal performance when looping over arrays, the order of the loops should be reversed in Julia relative to C/C++ (see relevant section of Performance Tips).

Julia values are not copied when assigned or passed to a function. If a function modifies an array, the changes will be visible in the caller.

In Julia, whitespace is significant, unlike C/C++, so care must be taken when adding/removing whitespace from a Julia program.

Ans 5.

- Julia is compiled, not interpreted
- Julia is interactive
- Julia has a straightforward syntax
- Julia combines the benefits of dynamic typing and static typing.
- Julia can call Python, C, and Fortran libraries
- Julia supports metaprogramming
- Julia has a full-featured debugger.
- Julia has a math-friendly syntax.
- Julia has automatic memory management
- Julia is developing its own native machine learning libraries