```
import Foundation

/**
 * MLPredictionOptions
 *
 * An object to hold options / controls / parameters of how
 * model prediction is performed
 */
@available macOS 10.13
open class MLPredictionOptions   NSObject

    /// Set to YES to force computation to be on the CPU only
    @available                          10.13               12.0
    open var usesCPUOnly  Bool

    /**
     * @abstract Propose the model to use the specified backing objects for the
     * output feature values.
     *
     * @discussion Use the property to get the inference result directly into the
     * client allocated buffer when possible for efficient memory management.
     *
     * The property is a dictionary of the feature name and the output backing
     * object.
     *
     * The framework may not use the specified backing object and instead
allocates
     * one by itself if the outputBacking dictionary doesn't contain the entry for
     * the feature name, the model doesn't support the user allocated buffers, or
in
     * the batch prediction mode. To check if the backing object was used,
compare
     * the output prediction and the backing object by object identity.
     *
     * \code
     *    CVPixelBufferRef outputBacking = ...;
     *    [options setOutputBackings:@{@"outputImage" : (__bridge
id)outputBacking}];
     *    id<MLFeatureProvider> prediction = [model
predictionFromFeatures:inputFeatures options:options error:&error];
     *    if ([prediction featureValueForName:@"outputImage"].imageBufferValue
== outputBacking) {
     *        // backing was used.
     *    }
     *    else {
     *        // backing was NOT used.
     *    }
     * \endcode
     *
     * The backing object must be either CVPixelBuffer or MLMultiArray
```

```
depending on
        * the feature value type.
        *
        * Do not lock the base address of the CVPixelBuffer. In the case of a
MLMultiArray
        * backed by a pixel buffer, make sure not to lock the underlying pixel buffer
by not
        * calling any data methods such as `.dataPointer` and subscript
methods before the
        * prediction.
        *
        * The framework ignores a backing object with an unknown feature name.
        *
        * For the best performance, use page-aligned address in MLMultiArray.
        *
        * \code
        *   #import <mach/vm_page_size.h>
        *   :
        *   void *backingBuffer = aligned_alloc(vm_page_size,
round_page(backingBufferSize));
        *   if (backingBuffer == NULL) { ... error handling ... }
        *   MLMultiArray *outputBacking = [[MLMultiArray alloc]
initWithDataPointer:(char *)backingBuffer
        *
...
        *
deallocator:^(void *) { free(backingBuffer); }
        *
... ];
        * \endcode
        *
        * For CVPixelBuffer backing, consider to use IOSurface-backed
CVPixelBuffer
        * created by CVPixelBufferPool because it is often the most efficient choice
for
        * memory footprint and performance, especially when the pixel buffers are
        * subsequently used for playback or export. (See also
AVSampleBufferDisplayLayer
        * and AVAssetWriter.)
        *
        * The output backing object must satisfy the output feature description's
        * `-isAllowedValue:` test, or the framework reporets an error at the
prediction
        * time. The exception is FP16 MLMultiArray backed by CVPixelBuffer, which
may be
        * accepted in Double or Float32 multi array output feature depending on the
        * underlying inference engine.
        *
        */
    @available macOS 11.0
    open var outputBackings   String   Any
```