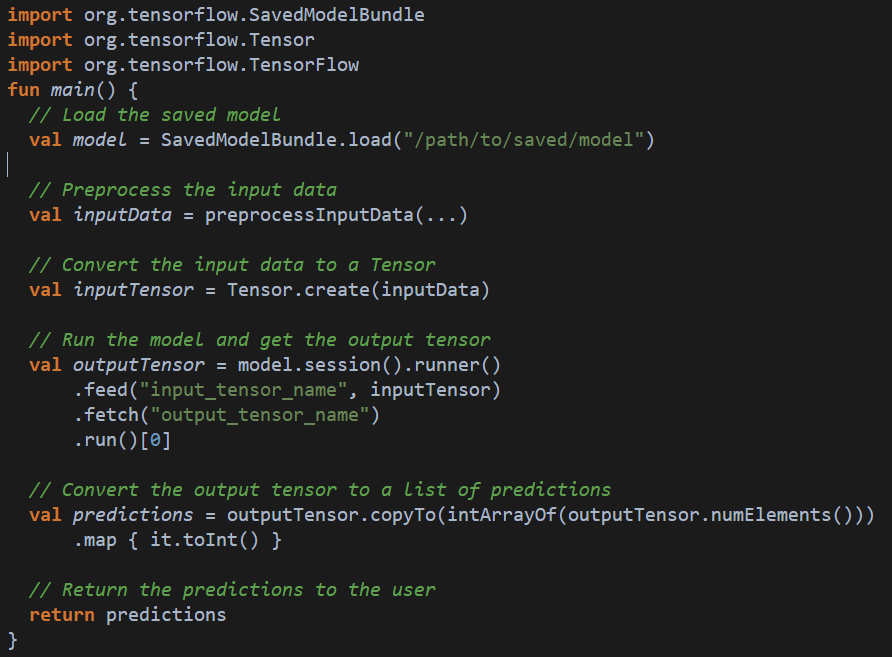
**Detect and verify people and objects using AI technology**

**Embedding the model directly into the app:**

**With Kotlin:**

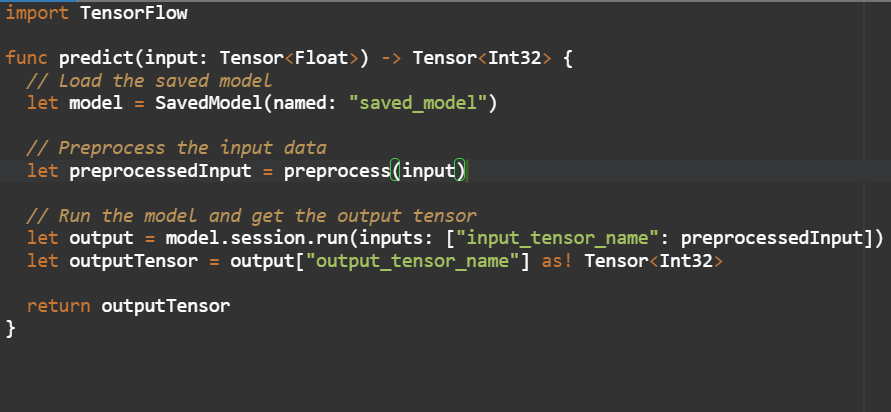
1. Save model: Once we have trained model, we will need to save it to a file so that we can load it later. We can use TensorFlow's **save** method or scikit-learn's **joblib.dump** function to do this.
2. Load model in Kotlin application: We can use TensorFlow's **load** method or scikit-learn's **joblib.load** function to load our saved model into our Kotlin application.
3. Preprocess input data: Before we can use our model to make predictions, we will need to preprocess our input data so that it is in the same format as the data our model was trained on. This may involve steps such as scaling, encoding categorical variables etc.
4. Use model to make predictions: Once our input data is preprocessed, we can use our model to make predictions by calling the appropriate method on our model object. For example, we can use TensorFlow's **predict** method or scikit-learn's **predict** function to make predictions with our model.
5. Return the predictions to the user: Finally, we will need to return the predictions to the user in some way, such as by displaying them on a Mobile Screen.

Here is an example of how we might write a Kotlin script to deploy a machine learning model in kotlin :



**With Swift:**

1. Save model: Once we have trained your model, we will need to save it to a file so that we can load it later. we can use TensorFlow's **save** method or scikit-learn's **joblib.dump** function to do this.
2. Load model in our Swift application: We can use TensorFlow's **SavedModel** class or scikit-learn's **joblib.load** function to load our saved model into our Swift application.
3. Preprocess input data: Before we can use our model to make predictions, we will need to preprocess our input data so that it is in the same format as the data our model was trained on. This may involve steps such as scaling, encoding categorical variables, or filling missing values.
4. Use model to make predictions: Once our input data is preprocessed, we can use our model to make predictions by calling the appropriate method on our model object. For example, we can use TensorFlow's **Session.run** method or scikit-learn's **predict** function to make predictions with our model.
5. Return the predictions to the user: Finally, we will need to return the predictions to the user in some way, such as by displaying them on a web page or returning them in an API response.



<https://www.tensorflow.org/swift/guide/overview>