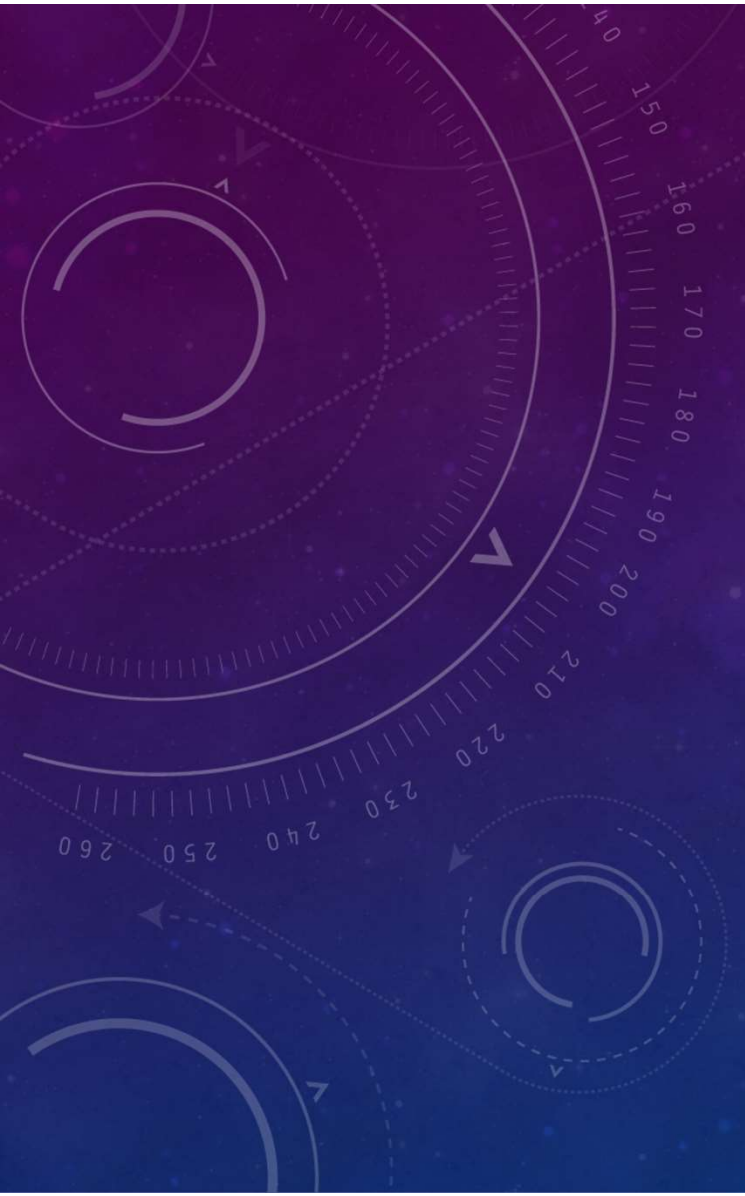

MACHINE LEARNING WITH .NET

Asmaliza Ahzan @ Emma

asmaliza@iverson.com.my

USING ML.NET WITH UWP

CHAPTER 10

BREAKING DOWN THE UWP ARCHITECTURE

- At a high level, UWP provides an easy framework to create rich desktop applications for Windows 10.
- As discussed, with .NET Core, UWP allows the targeting of x86, x64, and Advanced RISC Machine (ARM). At the time of this writing, ARM is not supported with ML.NET.
- In addition, UWP applications can also be written with JavaScript and HTML.
- A typical UWP desktop application includes the following core code elements:
  - Views
  - Models
  - View Models

MODEL-VIEW-VIEWMODEL (MVVM)

- Views, Models and View Models form a common app architecture principle of the Model-View-ViewModel, otherwise known as MVVM.
- In addition to the code components, images and audio are also common, depending on the nature of your application or game.
- Similarly, to mobile apps on the Android and iOS platforms, each app is sandboxed to specific permissions that you, the developer, request upon installation.
- Therefore, as you develop your own UWP applications, request only the required access that your app absolutely requires.

VIEWS

- Views, as we defined in the previous chapter's Blazor discussion, contain the user interface (UI) components of an application.
- Views in UWP development, such as those found in Windows Presentation Foundation (WPF) and Xamarin.Forms, use the Extensible Application Markup Language (XAML) syntax.
- The biggest differentiation between web development and UWP development is the powerful two-way binding XAML views when used with the MVVM principle.

MODELS

- Models provide the container of data between the View and View Model.
- Think of the Model as purely the transport for containing the data between the View and View Model.
- For example, if you had a movie list, a List collection of MovieItems would be defined in your MovieListingModel class.
- This container class would be instantiated and populated in the View Model, to be in turn bound in your View.

VIEW MODELS

- View Models provide the business-logic layer for populating your Model, and thereby your View indirectly.
- As mentioned previously, the MVVM binding provided in UWP development eases the management of trigger points to ensure your UI layer is up to date.
- This is achieved through the use of implementing the INotifyPropertyChanged interface in our View Model.
- For each property that we want to bind to our UI, we simply call OnPropertyChanged.
- The power behind this is that you can have complex forms with triggers within the setter of other properties, without having conditionals and endless code to handle the complexities.

CREATING THE WEB BROWSER CLASSIFICATION APPLICATION

- The application we will be creating is a web browser classification application.
- We will be using the SdcaLogisticRegression algorithm to take the text content of a web page, featurize the text, and provide a confidence level of maliciousness.
- In addition, we will be integrating this technique into a Windows 10 UWP application that mimics a web browser - effectively on navigation to a page - running the model and making a determination as to whether the page was malicious.
- If found to be malicious, we redirect to a warning page.

THE END