Challenge 3: First Steps on the Machine Learning Trail

Background

Adventure Works wants their data science team, including your team members, to begin learning how and when to perform custom machine learning with powerful, more programmatic APIs. Becoming proficient in machine learning takes some time, however, beginning with a high-level API, that is stable and has good documentation, is a great start. Some of the reasons to move on to custom machine learning include:

- To explore different algorithms to optimize what works best for the data
- To create a workflow with more control over your process
- To deploy a model to a specific architecture or configuration
- To overcome limits in bandwidth or data size for a service

In this challenge, *classical* (sometimes referred to as *statistical*) machine learning will be used. There are times when it makes sense to use, or at least begin with, a classical machine learning approach:

- There’s no need for expensive GPU
- Simple APIs enable fast prototyping
- This approach is used successfully in many production systems
- There’s support for a wide variety of machine learning algorithms

Prerequisites

- A development environment for sharing code and working in Jupyter
- The resized ***gear*** image data fom the previous challenges.
- An installation of the latest version of the **scikit-learn** Python package

Challenge

One of the most popular and well-established Python ML packages, **scikit-learn**, is often the go-to package for starting out, and is not uncommon in production systems. It has a simple, intuitive API (often modeled by other packages) and is a

great place to start for learning, implementing and programming traditional ML and basic neural networks in Python.

Your team must:

- Create a classification model to predict the class of a new gear image, training and validating it using the preprocessed gear image data.
- Calculate metrics with which to evaluate the model, including overall accuracy and a *confusion matrix* that indicates the level of correct and incorrect predictions for each class.
- Use the model to predict the class of at least five relevant images that are not included in the ***gear*** dataset. You can find these images by using Bing to search for appropriate terms, for example:
  - https://www.bing.com/images/search?q=ski+helmet
  - https://www.bing.com/images/search?q=climbing+axe
  - https://www.bing.com/images/search?q=tent
  - https://www.bing.com/images/search?q=carabiner
  - https://www.bing.com/images/search?q=insulated+jacket

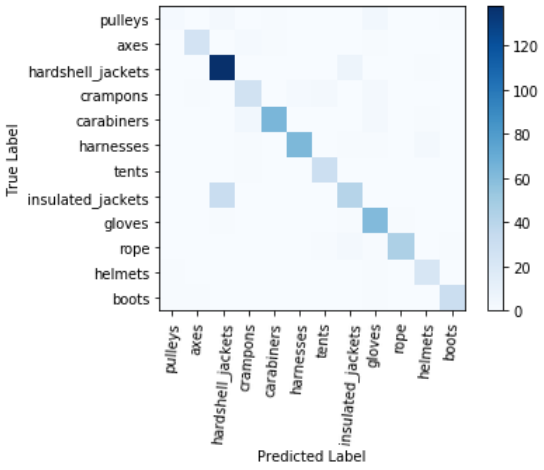
Hints

- Use the **pip** command to install or upgrade the **scikit-learn** library in your Python environment. You can use `pip -h` for help with the syntax.
- You can use the `!` prefix to run `pip` in a notebook cell. If you are using the Data Science Virtual Machine (on which there are mulitple Python environments), you may need to use the `sys.executable` module to do this as explained in the DSVM documentation.
- Split the processed data into *train* and *test* data sets with which to train a machine learning model and calculate its accuracy. To optimize model quality, the training data set is usually much larger than the test set while still leaving enough data to adequately test the model.
- Use the *test* dataset to calculate metrics that will enable you to evaluate your model. Key metrics include overall *accuracy* and a *confusion matrix*. The **scikit-learn** library includes functionality to calculate these, as well as additional metrics such as *precision* and *recall*.
- Try multiple classification algorithms from the **scikit-learn** library and compare the results. *Logistic Regression* and *Decision Tree* techniques are a common starting point.
- When using your model to predict the class of a new image, apply the same pre-processing to the new image that you applied to the data used to train the model.

Success Criteria

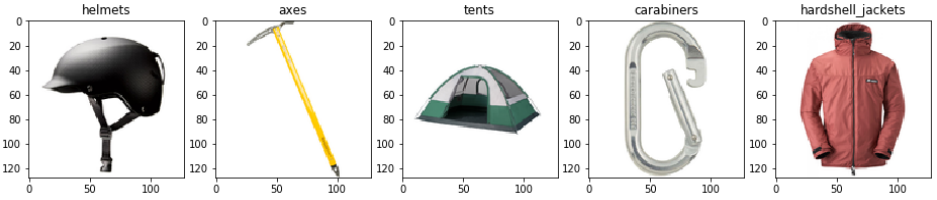
- Achieve an *accuracy* score from your model based on the *test* data set of **0.80**(80%) or greater.

- Show a *confusion matrix* that indicates more correct predictions than incorrect predictions for each class in your *test* data set. For example:



Confusion matrix

- Show predictions for the five images you found in the Challenge section, like this:



Gear predictions

(Note: Your model is not required to predict the correct class for all of the images, but it would be good if it does!)

References

- An introduction to machine learning with scikit-learn
- Supervised learning
- Supervised learning algorithms in scikit-learn
- Model evaluation