1

Introduction to Machine Learning with Keras

Present 1-3: The Title slide and the Learning Objectives slide. An overview of what we will achieve in this course.

Lesson Objectives

By the end of this lesson, you will be able to:

* Revisit Machine Learning with scikit-learn
* Understand how to preprocess data for a machine learning model
* Build a Logistic Regression Model with scikit-learn

Introduction

Announce: Welcome the class. Introduce yourself and discuss what the course will cover. Talk about the topics that will be covered in this lesson.

Machine learning is the science of utilizing machines to emulate human tasks and to have the machine improve in performance of that task over time. By feeding machines data in the form of observations of real-world events they can develop patterns and relationships that will optimize an objective function, such as the accuracy of a binary classification task. In general, the usefulness of machine learning comes in the ability to learn highly complex and nonlinear relationships in large datasets and to replicate the results of that learning many times. Take for example the classification of a dataset of pictures of either dogs or cats into classes of their respective type. For a human this is trivial and accuracy would likely be very high, however it may take around a second to categorize each picture and scaling the task can only be achieved by increasing the number of humans, which may be infeasible. While it may be difficult to reach the same level of accuracy for this task for machines, though certainly not impossible, the advantages of machines are that they can classify many images per second, and scaling is as simple as copying code from one machine to another.

In this course we are using the programming language Python and the libraries scikit-learn and keras to create machine learning models. We are using Python because of its broad level of utilization across the machine learning community and easy learning curve for new users. Much the same, scikit-learn is well established within the community and flexible enough to create a wide variety of machine learning models in a simple-to-use API that enables beginners to efficiently prototype without having to have a deep understanding or code each specific model. Moreover, scikit-learn is open source and well documented so there is a community of active participants working to improve the library.

Due to its ease of use, flexibility, and good support scikit-learn has been adopted as the machine learning library from beginners to industry practitioners alike. However, while scikit learn is certainly excellent for as a library for learning machine learning in general the estimators that it provides for neural networks are lacking. One big reason for this is that there is much more flexibility in the architecture of neural networks that is antithesis to scikit-learn. Neural networks can exhibit a number of different architecture designs, with various optimizers, activation functions, and layer properties, these terms will become more familiar as the course progresses. In order for scikit-learn to incorporate neural networks it may well double the size of the library entirely so instead the creators thought it be best to let another library tackle the richness of neural networks.

Similar to scikit-learn, keras makes it easy to create models in the Python programming language through an easy-to-use API. However, the goal of keras is for the creation and training of neural networks, rather than machine learning models in general. The library has much of the general-purpose functions built in such as optimizers, activation functions and layer properties such that users, like with scikit-learn, do not have to code these algorithms from scratch.

While this course is introduction to deep learning with keras we will begin with a lesson in scikit-learn to establish the fundamentals of building a machine learning model using the python programming language. Many of the steps to create models are highly transferable between the two libraries and scikit-learn has the advantage of being widely used and as such there is much documentation, tutorials, and learning to be found across the internet. Specifically, by the end of this lesson students will be able to create a logistic regression model utilizing the scikit-learn library. The fundamentals of preprocessing the data from scratch, optimization techniques, and model evaluation steps will be covered in this lesson. In subsequent lessons we will develop the same logistic regression model in the keras library by building a neural network.

Scikit-learn

Present: Slide introducing scikit-learn

Scikit-learn was initially created in 2007 as a way to easily create machine learning models in the Python programming language by David Cournapeau. Since its inception, the library has grown immensely in popularity because of its ease-of-use, wide adoption within the machine learning community and flexibility of use.

Note

You can find all the documentation for the scikit-learn library here: <https://scikit-learn.org/stable/documentation.html>.

The estimators in scikit-learn can generally be classified into supervised learning and unsupervised learning techniques. Supervised learning occurs when the target variable is known and models are trained in order to correctly predict the target variable. Binary classification using logistic regression is a good example of a supervised learning technique. In unsupervised learning there is no target variable given in the training data but models aim to assign a target variable. An example of an unsupervised learning technique is k-means clustering in which data is partitioned into a given number of clusters based on proximity to neighboring data points, the target variable assigned may be either the cluster number or cluster center.

**Note**

Some more information on the difference between the two forms can be found here: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

There are even exist semi-supervised learning techniques in which unlabeled data is used in the training of machine learning models. This may occur if there is only a small amount of labelled data and a copious amount of unlabeled, in practice the unlabeled data coupled with labelled data produced significant improvement in model performance as compared to unsupervised learning, and in instances where labelling the unlabeled data is not feasible.

The scikit-learn library is ideal for beginners as the general concepts for building machine learning pipelines can be learned easily. Concepts such as data preprocessing, hyperparameter tuning, and model evaluation, and many more are all taken care of in the library. Even experienced users find the library easy to rapidly prototype models before using a more specialized machine learning library.

Indeed, the various machine learning techniques discussed such as supervised and unsupervised learning can be applied with keras using neural networks with different architectures that will be discussed throughout the course

Discuss: Discuss with students their familiarity with the library

Keras

Present: Slide introducing keras

Created in 200X by Y built to make deep learning accessible to everyone. Keras is designed to be a high-level neural network API that is built on top of the frameworks Tensorflow, CNTK, or Theano. One of the great benefits of using Keras as an introduction to deep learning for beginners is that it is very user-friendly-- advanced functions such as optimizers, layers, etc, are already built in to the library and do not have to be written from scratch. In fact, this is why keras is so popular not only amongst beginners but seasoned experts also, the library allows for rapid prototyping of neural networks, supports a wide variety of network architectures and is can be run on both CPU and GPU.

Note

You can find the library and all documentation for keras at the following link: https://keras.io/.

Discuss: Discuss other deep learning frameworks with the students, including Tensorflow, pytorch and MXNet, so that students understand the potential of keras as beginners, and understand the benefits and pitfalls of other frameworks such as infrastructure compatibility.

About the Library

Present: This slide shows the various uses of keras

Keras is best used to create and train neural networks and does offer much in terms of other machine learning algorithms such as supervised algorithms like support vector machines or unsupervised algorithms like k-means clustering. What keras does offer, though, is a well-designed API to create and train neural networks, that takes away much of the effort to apply the linear algebra and multivariate calculus correctly.

Note

Almost all the math required to understand the algorithms behind deep learning are linear algebra and mulitvariate calculus. For more information on this, visit: <https://towardsdatascience.com/deep-learning-basic-mathematics-for-deep-learning-a82981e95e3b>. For those more mathematically capable you can visit: https://towardsdatascience.com/https-medium-com-piotr-skalski92-deep-dive-into-deep-networks-math-17660bc376ba

The specific modules available from the keras library, such as neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes will be explained deeply throughout the course. All these modules have relevant functions that can be used to optimize the performance for training neural networks for specific tasks.

Announce: Announce where documentation can be found

Advantages and Disadvantages

Present: Slide showing the advantages and disadvantages of keras

Presented below are a few of the main advantages and disadvantages of using keras for machine learning purposes.

Advantages

* User friendliness

Much like scikit-learn, keras features an easy-to-use API, that allows users to focus on model-building rather than the specifics of the algorithms.

* Modular

The API consists of fully-configurable modules that can all be plugged together and that work seamlessly.

* Extensible

It is facile to add new modules to the library. This allows users to take advantage of the many robust modules within library while also providing them the flexibility to create their own.

* Open Source

Keras is an open source library and is constantly improving and adding modules to its codebase thanks to the work of many collaborators working in conjunction to build improvements and help create a robust library for all.

* Works with Python

Keras models are declared directly in Python rather than in separate configuration files which allows keras to take advantages of working with python such as ease of debugging and extensibility

Disadvantages

* Advanced customization

While simple surface level customization such as creating simple custom loss functions or neural layers is facile, it can be difficult to change how the underlying architecture works.

* Lack of examples

Beginners often rely on examples to kickstart their learning. Advanced examples can be lacking in keras’ documentation that can prevent beginners from advancing in their learning.

Keras offers those familiar with the Python programming language and machine learning experience the ability to create neural network architectures easily. Though because neural networks themselves are quite complicated we will begin with scikit-learn to introduce many machine learning concepts before trying them out in the keras library

Discuss: Discuss other machine learning libraries with the students, and their specialties.

More than building models

Present: This slide shows the various uses of machine learning libraries other than building models

While machine learning libraries such as scikit-learn and keras were created as a method to build and train predictive models, their practicality extends much further beyond.

Discuss: Discuss the multitude of areas and use cases that students think machine learning has made an impact.

One common use case of building models is that they can be utilized to perform predictions on new data. Once a model is trained new observations can be fed into the model to generate predictions from the new observations. Another common use case for models is that they can be used to summarize datasets by learning representations of the data. An example is that autoencoders, a type of neural network architecture, can be used to learn such representations of a given dataset so that it can be represented in a reduced dimension with minimal loss of information. Another popular use of models is to

Data Representation

Present: Slide introducing the new topic

Discuss: Discuss the common formats of data, how variables can be represented, and what is required for amchine learning models.

We build models so that we can learn something about the data we are training on, and about the relationships between the features that can hopefully inform us when we encounter new observations. However, we must realize that the observations we interact with in the world and the format of data needed for the algebra used to train the models to work are very different. Working with text data is a prime example of this, when we read text, we are able to understand the text, apply context, and the make sense of the order of the words that make up the piece of text. However, machine are not able to pick up on much of this contextual information unless it specifically encoded, they have no idea how to convert text into something that be an input to a matrix to apply multivariate calculus. Therefore, we must represent the data appropriately, often by converting non-numerical data types. Such as text, dates, and categorical variables into numerical ones.

Tables of Data

Present: This slide shows how real-world data is often represented

Many data fed into machine learning problems are two-dimensional, and can be represented as rows or columns, images are a good example of a dataset that may be three or even four-dimensional, the shape of each image will be two dimensional, (a height and width), many images together will add a third dimension, and a color channel (i.e. red, green, blue) will add a fourth.

The table below is a sample of a dataset taken from UCI’s machine learning repository. The dataset describes marketing campaign results of a Portuguese banking institution. One objective of the dataset would be to predict whether or not a customer subscribed to the particular product given features about each customer and how they were observed to interact with the marketing campaign. In the dataset each customer is designated a row and the dataset is labeled whether or not the customer subscribed.

Discuss: Is this a supervised or unsupervised machine learning problem

Announce: Wide variety of open source data that is available on UCI machine learning repository

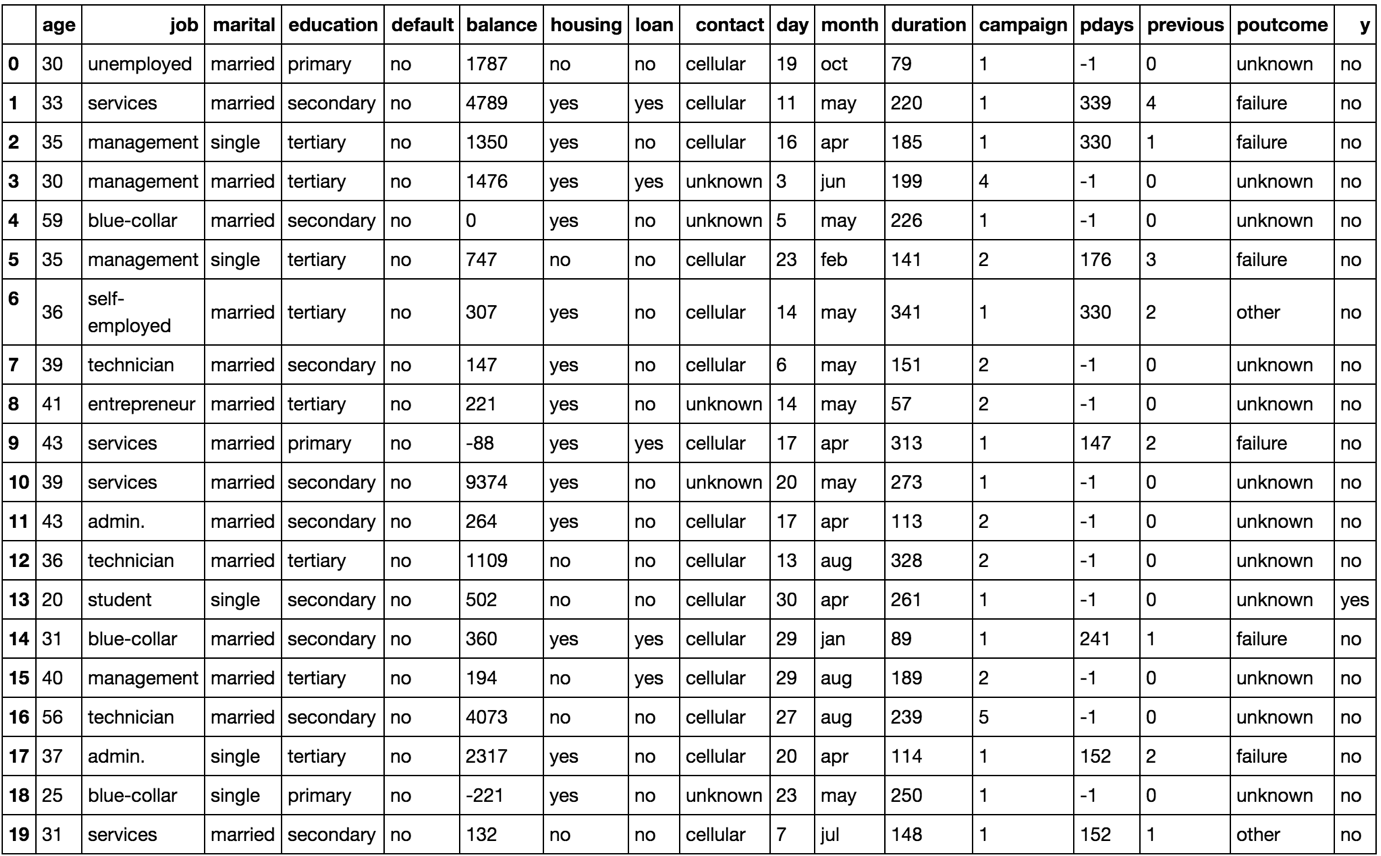


Figure 1.1: An image showing the first 20 instances of the marketing dataset

Loading in data

Present: This slide explains how to load in data

Data can take many forms and can be available from many places. Datasets for beginners are often given in a flat format, which means that they are 2 dimensional, having rows and columns. Other common forms of data may include images, JSON objects, text documents, amongst others. Each type of data format has to be loaded in specific ways. For example, numerical data can be loaded into memory using the NumPy library, which is an efficient library for working with matrices in Python. However, we would not be able to load our marketing data csv into memory using the NumPy library because the dataset contains string values. For our dataset we will use the Pandas library because of its ability to easily work with various data types such as strings, integers, floats, and binary values. In fact, Pandas is dependent on NumPy for operational on numerical data types. Pandas is also able to read JSON, excel documents, and database using SQL queries, which makes the library common amongst practitioners for loading and manipulating data in Python.

Other forms of data common in deep learning such as images and text will be discussed later in the course.

Discuss: Discuss amongst students possible sources of data

Exercise 1: Loading a sample dataset from the UCI machine learning repository

Note

All exercises and activities will be primarily developed in Jupyter Notebook. It is recommended to keep a separate notebook for different assignments, unless advised not to.

In this exercise, we will be loading the bank marketing dataset from the UCI machine learning repository.

Note

For the exercises and activities within this lesson, you will need to have Python 3.6, Jupyter and Pandas installed on your system.

1. Open a Jupyter Notebook to implement this exercise.

In the cmd or terminal, navigate to the desired path and use the following command: jupyter notebook

1. Download the zip file from the website <https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.zip>. This can be achieved either in the cmd, or terminal, but since we have the notebook open we can run our shell commands in a notebook cell:

!wget https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.zip

1. Now that contents have been downloaded we will have to uncompress the contents to use them, let’s make a separate folder and unzip the contents of the zipped file into the folder.

!mkdir data

!unzip bank.zip -d data/

Note

The -d parameter is an argument to unzip the contents of the file into the given folder.

1. To verify the data looks as follows we can look at the first 10 rows of the csv using the head function.

!head data/bank.csv

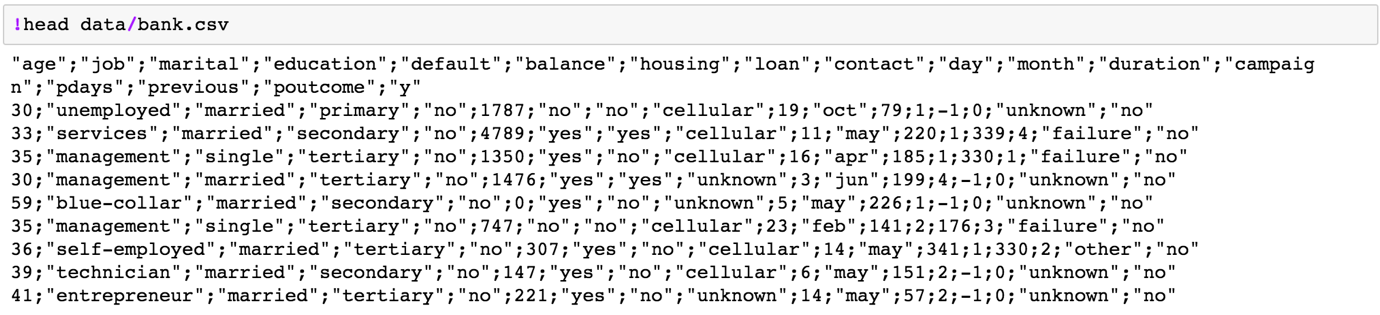


Figure 1.3: A screenshot showing the first 10 rows of the csv

1. Now let’s load the data inot memory using the Pandas library with the read\_csv. function, first importing the pandas library.

import pandas as pd

bank\_data = pd.load\_csv(‘data/bank\_data.csv’, sep=’;’)

1. Finally, to verify we have the data loaded into memory corectly we can print the first rows.

Then, print out the top 10 values of the variable.

bank\_data.head(20)

Y.head(10)

The printed output should look as below:

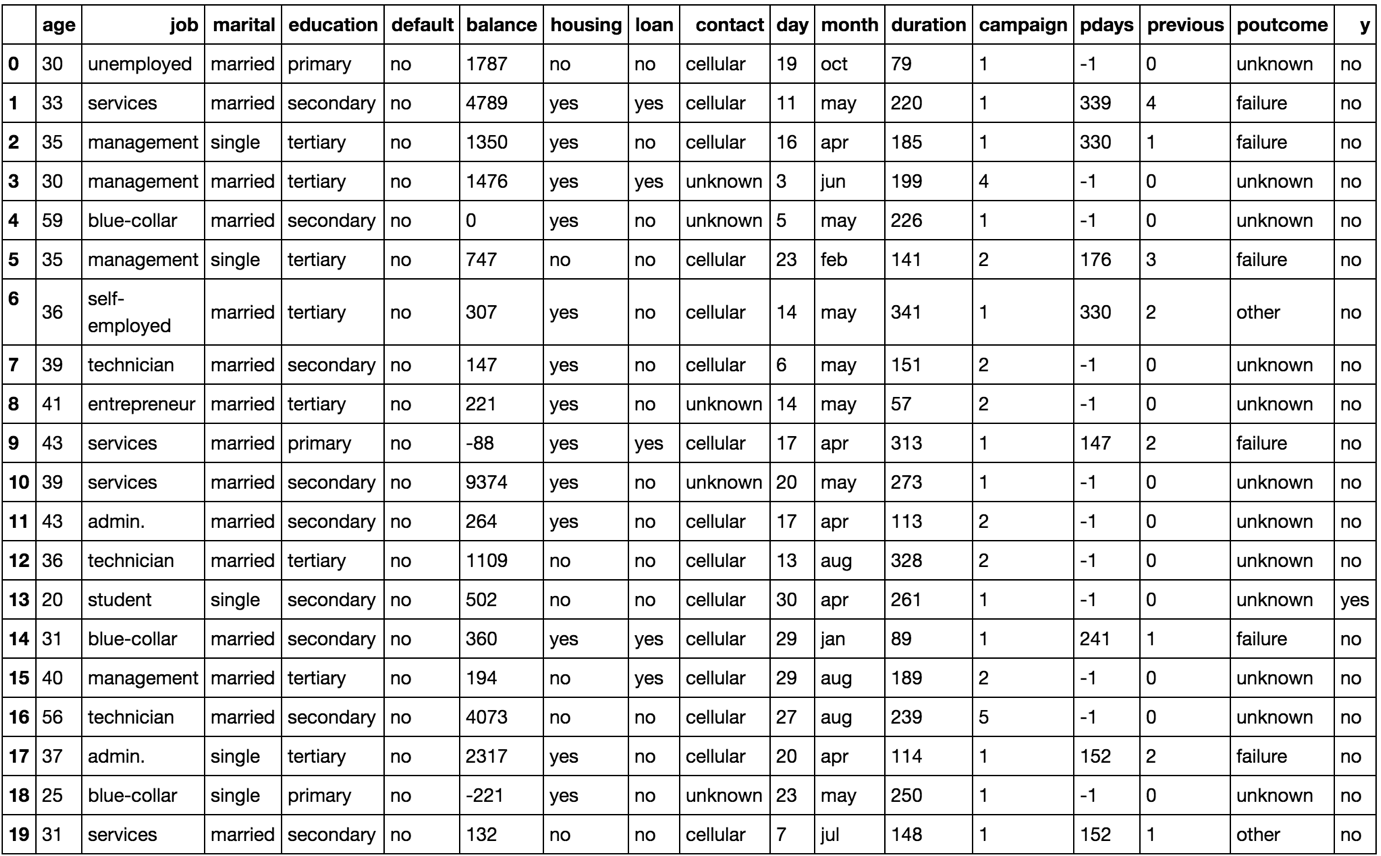


Figure 1.4: A screenshot showing the first 20 rows of the pandas dataframe

We can also print the shape of the dataframe.

bank\_data.shape

The printed output should look as below, showing that the dataframe has 4521 rows and 17 columns:

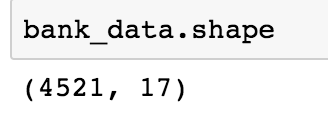


Figure 1.5: A screenshot showing the output of the shape command on the dataframe

Now we have successfully loaded the data into memory we can begin to manipulate and clean the data such that a model can be trained using the data. Remember that machine learning models require data to be represented as numerical data types to be trained. We can see from the first few rows of the dataset that some of the columns are string types, so we will have to convert them to numerical data types later in the lesson.