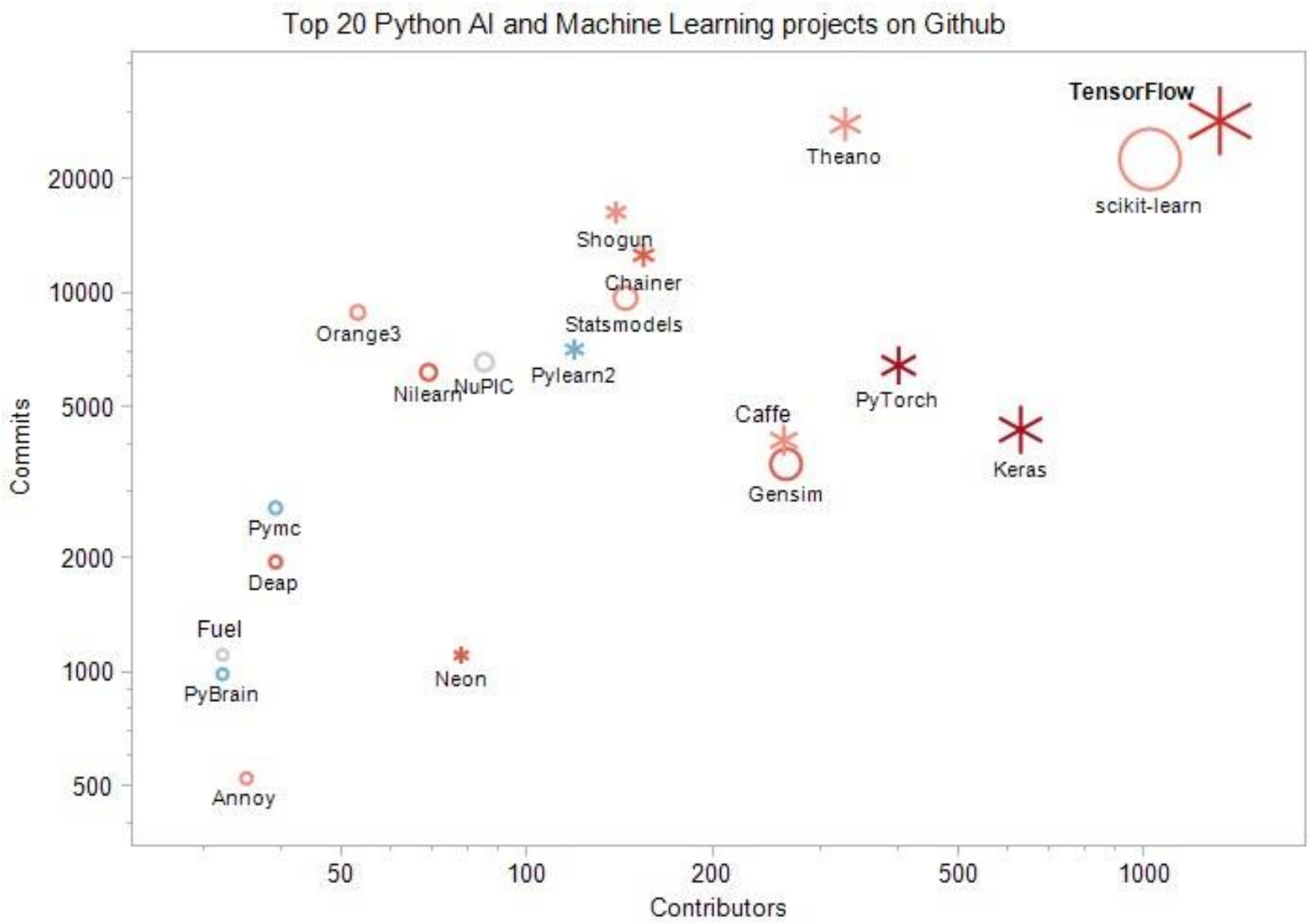
Topic 3
Python Machine Learning

ST0249 (AIML) AI & MACHINE LEARNING

Learning Outcomes

- Use of Python Machine Learning Tools
  - Describe some available python machine learning tools
  - Install and configure Scikit-learn
- Understand Model Evaluation Techniques
  - Explain train set
  - Explain test set
- Understand Evaluation Metrics and Scoring
  - Understand use of evaluation metrics in model selection

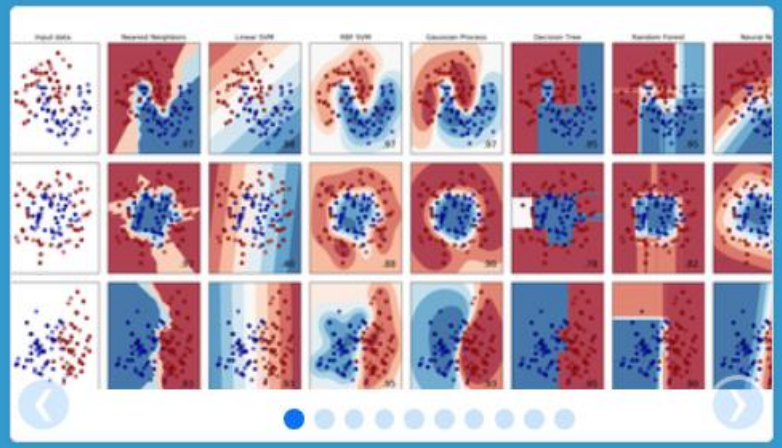
Use of Python Machine Learning Tools



source https://www.kdnuggets.com/2018/02/top-20-python-ai-machine-learning-open-source-projects.html

Open source ML

- **Scikit-learn** is simple and efficient tools for data mining and data analysis, accessible to everybody, and reusable in various context, built on NumPy, SciPy, and matplotlib, open source, commercially usable – BSD license.
  - https://github.com/scikit-learn/scikit-learn
  - http://scikit-learn.org/stable/



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

What does scikit-learn do?

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

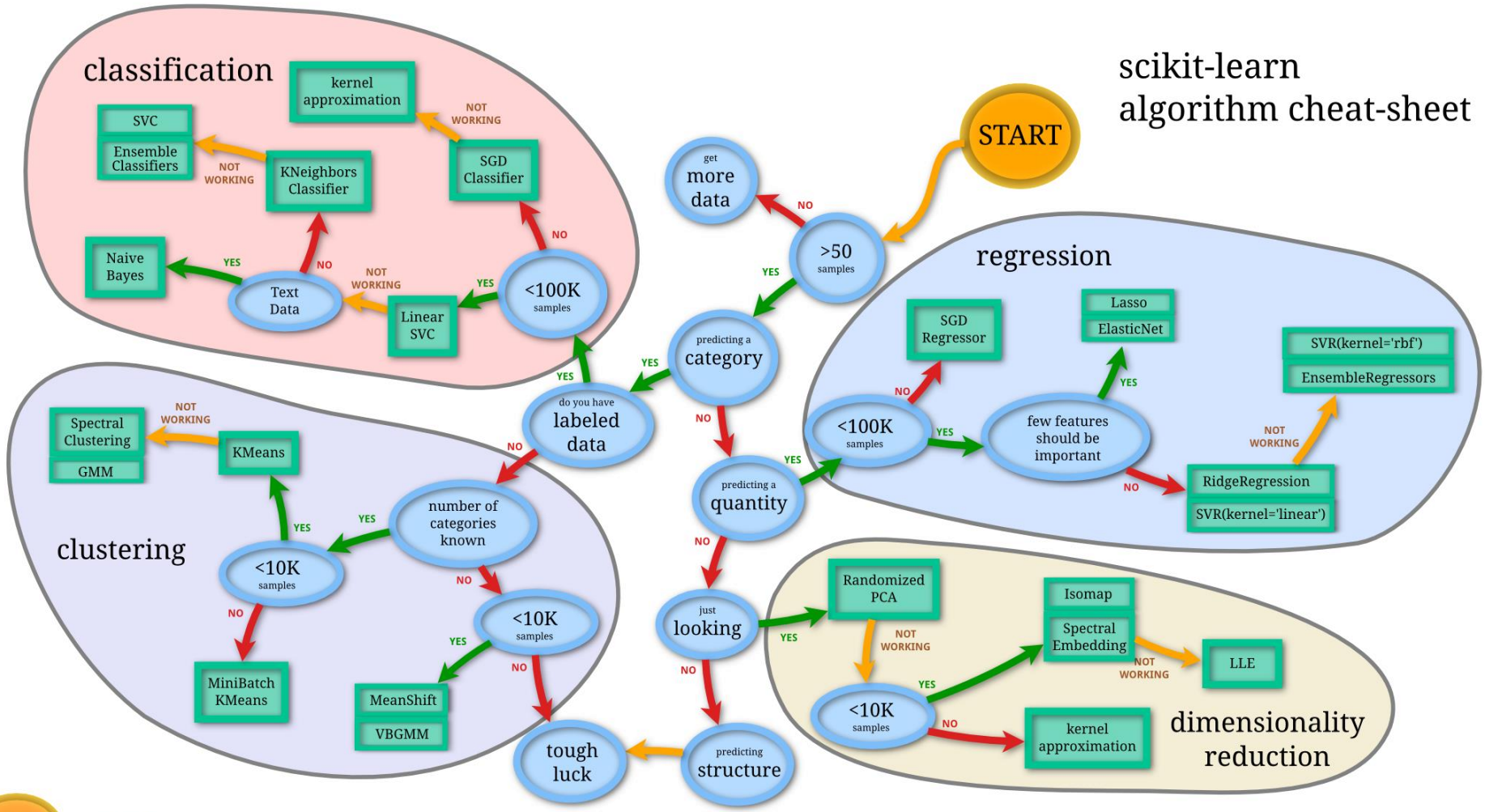
Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples



source http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit-learn installation

http://scikit-learn.org/stable/install.html

Installing the latest release

Scikit-learn requires:

- Python (>= 2.7 or >= 3.3),
- NumPy (>= 1.8.2),
- SciPy (>= 0.13.3).

If you already have a working installation of numpy and scipy, the easiest way to install scikit-learn is using `pip`

```
pip install -U scikit-learn
```

or `conda`:

```
conda install scikit-learn
```

Anaconda ships with a recent version of scikit-learn, in addition to a large set of scientific python library for Windows, Mac OSX and Linux.

Scikit-learn documentation

Scikit-learn is extensive documented. The user guide document is available as a download from:
http://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf

There is also an online version at:
http://scikit-learn.org/stable/user_guide.html

Scikit-learn datasets

scikit-learn comes with a few small standard datasets that do not require to download any file from some external website.

These datasets are useful to quickly illustrate the behavior of the various algorithms implemented in the scikit. They are however often too small to be representative of real world machine learning tasks.

load_boston([return_X_y])	Load and return the boston house-prices dataset (regression).
load_iris([return_X_y])	Load and return the iris dataset (classification).
load_diabetes([return_X_y])	Load and return the diabetes dataset (regression).
load_digits([n_class, return_X_y])	Load and return the digits dataset (classification).
load_linnerud([return_X_y])	Load and return the linnerud dataset (multivariate regression).
load_wine([return_X_y])	Load and return the wine dataset (classification).
load_breast_cancer([return_X_y])	Load and return the breast cancer Wisconsin dataset (classification).

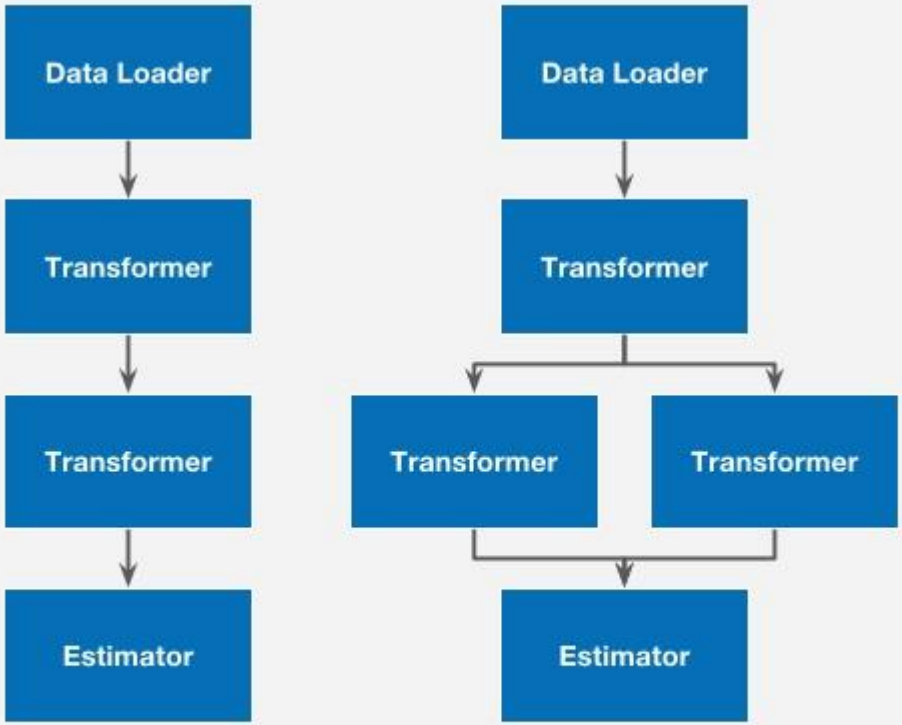
source http://scikit-learn.org/stable/datasets/index.html

Scikit-learn Development Environment

- Unlike Azure ML Studio, scikit-learn does not come with an integrated development environment or graphical ML workflow tool.
- You can use the development environment launched from Anaconda; such as Spyder or Jupyter notebook.
- Azure ML Studio is able to support Python Scripting. You can run the the Scikit-learn within the Azure ML Studio Python Scripting Module. Hence, you can use Azure ML Studio a placeholder integrated development environment for Scikit-learn. You can deploy the running scikit-learn code as part of the running ML experiment into a Azure ML Web Service.

Scikit-learn pipeline

- Pipelines are an extremely simple yet very useful tool for managing machine learning workflows.
  - A typical machine learning task generally involves data preparation to varying degrees. Such tasks are known for taking up a large proportion of time spent on any given machine learning task.
  - After a dataset is cleaned up from a potential initial state of massive disarray, however, there are still several less-intensive yet no less-important transformative data pre-processing steps such as feature extraction, feature scaling, and dimensionality reduction, to name just a few.
  - Maybe your pre-processing requires only one of these transformations, such as some form of scaling. But maybe you need to string a number of transformations together, and ultimately finish off with an estimator of some sort.



Scikit-Learn Pipelines: `fit()` and `predict()`

Scikit-learn Pipelines

- Scikit-learn's Pipeline class is designed as a manageable way to apply a series of data transformations followed by the application of an estimator. It is a pipeline of transforms with a final estimator.
- Pipelines is useful for:
  - Convenience in creating a coherent and easy-to-understand workflow
  - Enforcing workflow implementation and the desired order of step applications
  - Reproducibility
  - Value in persistence of entire pipeline objects

Understand Model Evaluation Techniques

Model evaluation: checking if the algorithm/model learnt from the data is good enough to be put to use

- The key idea is that the algorithm/model that has been trained should be able to make good predictions for new data.
- In the absence of new data during development, we set aside some data from the original dataset as the test dataset. The test data stands in as a proxy for future new data.
- The remaining data (training dataset) is used to build/train the model.
- We will calculate the "goodness" of the prediction using metrics (see next section).
- The metrics are calculated for both the training and test datasets.

Understand Evaluation Metrics and Scoring

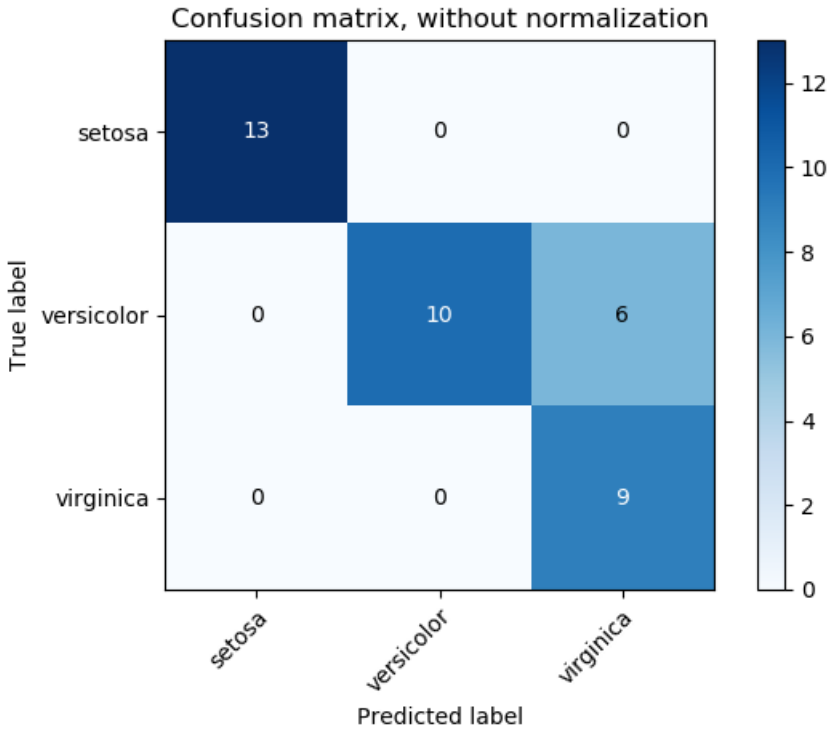
Model evaluation for classifiers

- The metrics used for classification are totally different from those used regression.
- The metrics for classifiers measure that amount of misclassification (i.e. assigning the wrong label to the output) that happens.
- In scikit-learn, the metrics for regression are found at https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
- The accuracy_score function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions.
- The best possible score is 1.0 (100% accurate). The worse is 0.5 which corresponds to random guessing. Score of 0.0 mean there is mislabelling of the correct answer since being 100% wrong is not possible without a certain degree of correct scoring/prediction.

Metrics for Classification: Confusion Matrix

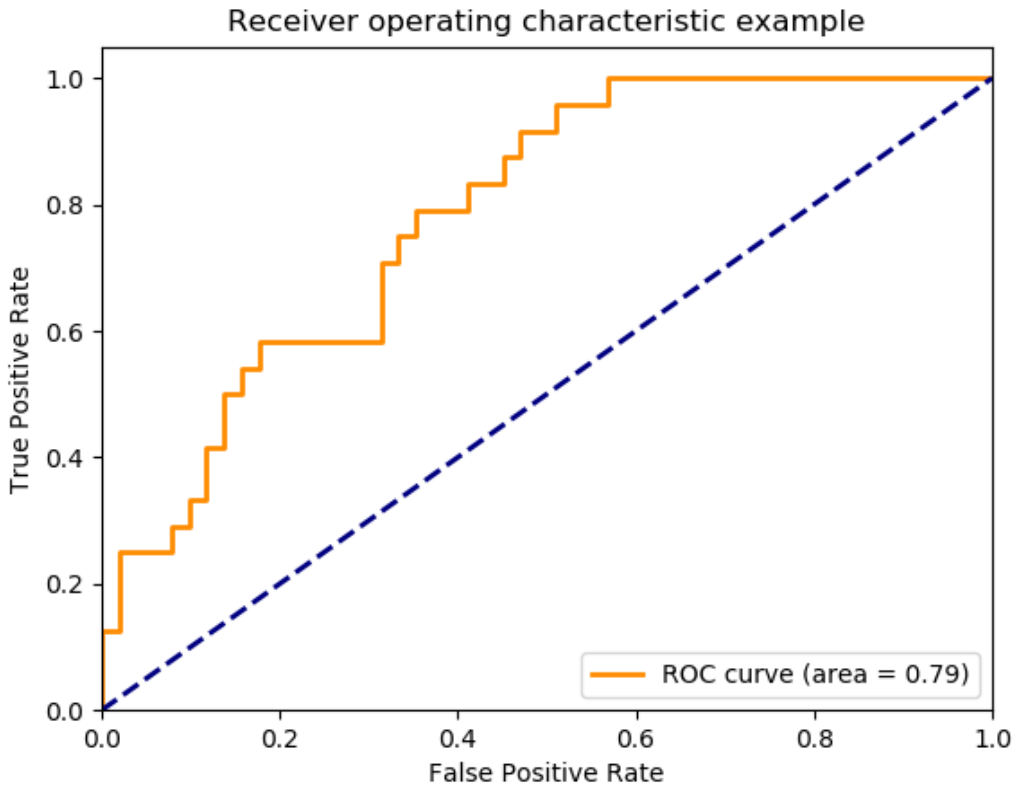
https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix

The confusion_matrix function evaluates classification accuracy by computing the confusion matrix with each row corresponding to the true class



Metrics for two-class/binary classification: ROC curve

https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics



Metrics for two-class/binary classification: ROC curve

A receiver operating characteristic (ROC), or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the positives (TPR = true positive rate) vs. the fraction of false positives out of the negatives (FPR = false positive rate), at various threshold settings. TPR is also known as sensitivity, and FPR is one minus the specificity or true negative rate.

Metrics for two-class/binary classifier: Area Under ROC Curve (AUC)

The roc_auc_score function computes the area under the receiver operating characteristic (ROC) curve, which is also denoted by AUC or AUROC. By computing the area under the roc curve, the curve information is summarized in one number.

Best possible AUC is 1.0.

Worst AUC is 0.5 (random guessing)

Model evaluation for regressors

- The metrics used for regression are totally different from those used classification.
- The metrics used for regression basically measure the amount of error in the predictions by comparing the predicted value/score (yhat) with the actual value of the lable (y).
- The Residuals (error) e = y – yhat
- In scikit-learn, the metrics for regression are found at https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics
- The r2_score function computes the coefficient of determination, usually denoted as R^2. The best possible r2_score (perfect) is 1.0.
- r2_score can be negative (because the model can be arbitrarily worse). A constant model, disregarding the input features, would get a R^2 score of 0.0.

Metrics for Regression

Mean squared error	MSE	=	$\frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	RMSE	=	$\sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	MAE	=	$\frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	MAPE	=	$\frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Summary

We have learnt that:

- Scikit-learn is a python library that contains many different implementations of machine learning (ML) algorithms
- Scikit-learn also has many helper/utility routines machine learning such as encoding categorical variables (pre-processing)
- Scikit-learn has routines for evaluating the results of the modelling process (model selection)
- Scikit-learn has some sample data sets included in the library