Advanced  python™ Packages

Miguel Pereira

Outline

3 python packages for data science:

1. Scikit-learn
   https://scikit-learn.org/

2. Tensorflow
   https://www.tensorflow.org/

3. Pytorch-geometric
   https://pytorch-geometric.readthedocs.io/

Structure

For each package:

- What is does?
- Methods implemented
- Code example on Google Colaboratory
- Where to find more information and resources?

Goals

1. Provide a taster for each package and its capabilities
2. Jumpstart learning (‘the spark’)
3. Getting comfortable with the documentation interface on the official websites

Disclaimer: This does not replace all the hard work.


scikit
learn

scikit-learn

Machine Learning in Python

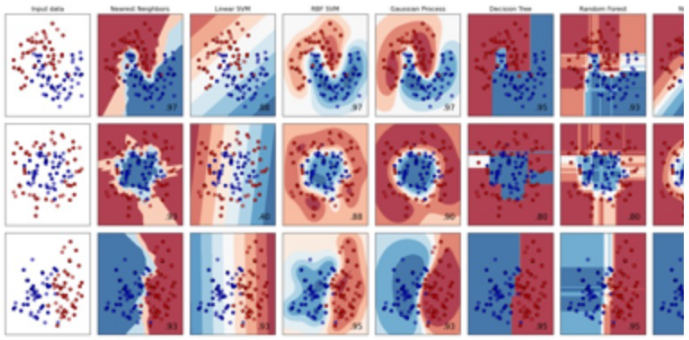
Getting Started | Release Highlights for 0.24 | GitHub

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

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

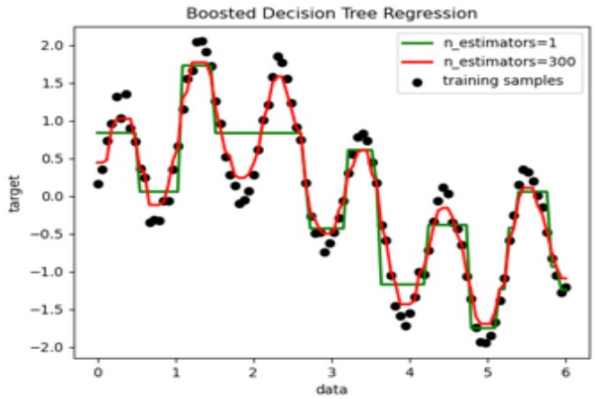


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

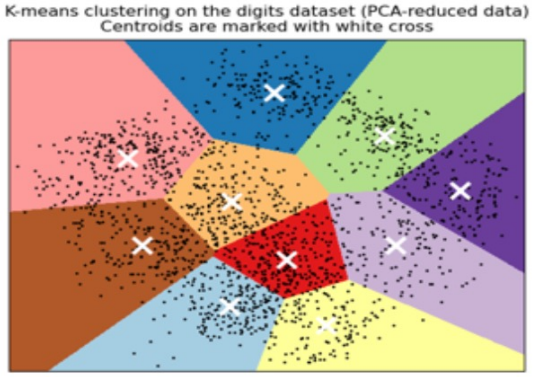


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

What it does?

- Scikit-learn is:
  - All-in-one toolbox for machine learning in Python
  - Built on Scipy, Numpy and matplotlib
  - Integrates seamlessly with Pandas
  - Includes:
    - Data processing tools
    - Data modelling functions
    - Model training support tools
    - Model explainability tools
    - Native plotting support
    - Built-in datasets
    - Model ‘persistence’ (save and load model objects)
    - Basis for many third-party libraries that extend scikit-learn’s capabilities (e.g. ELI5, scikit-survival)

Data processing functions

- Data Normalisation
- Creation of dummy variables
- Missing data imputation

`sklearn.preprocessing`: Preprocessing and Normalization

The `sklearn.preprocessing` module includes scaling, centering, normalization, binarization methods.

User guide: See the Preprocessing data section for further details.

`preprocessing.Binarizer`(*[, threshold, copy])	Binarize data (set feature values to 0 or 1) according to a threshold.
`preprocessing.FunctionTransformer`([func, ...])	Constructs a transformer from an arbitrary callable.
`preprocessing.KBinsDiscretizer`([n_bins, ...])	Bin continuous data into intervals.
`preprocessing.KernelCenterer`()	Center a kernel matrix.
`preprocessing.LabelBinarizer`(*[, neg_label, ...])	Binarize labels in a one-vs-all fashion.
`preprocessing.LabelEncoder`()	Encode target labels with value between 0 and n_classes-1.
`preprocessing.MultiLabelBinarizer`(*[, ...])	Transform between iterable of iterables and a multilabel format.
`preprocessing.MaxAbsScaler`(*[, copy])	Scale each feature by its maximum absolute value.
`preprocessing.MinMaxScaler`([feature_range, ...])	Transform features by scaling each feature to a given range.
`preprocessing.Normalizer`([norm, copy])	Normalize samples individually to unit norm.
`preprocessing.OneHotEncoder`(*[, categories, ...])	Encode categorical features as a one-hot numeric array.
`preprocessing.OrdinalEncoder`(*[, ...])	Encode categorical features as an integer array.
`preprocessing.PolynomialFeatures`([degree, ...])	Generate polynomial and interaction features.
`preprocessing.PowerTransformer`([method, ...])	Apply a power transform featurewise to make data more Gaussian-like.
`preprocessing.QuantileTransformer`(*[, ...])	Transform features using quantiles information.
`preprocessing.RobustScaler`(*[, ...])	Scale features using statistics that are robust to outliers.
`preprocessing.StandardScaler`(*[, copy, ...])	Standardize features by removing the mean and scaling to unit variance

`preprocessing.add_dummy_feature`(X[, value])	Augment dataset with an additional dummy feature.
`preprocessing.binarize`(X, *[, threshold, copy])	Boolean thresholding of array-like or scipy.sparse matrix.
`preprocessing.label_binarize`(y, *, classes)	Binarize labels in a one-vs-all fashion.
`preprocessing.maxabs_scale`(X, *[, axis, copy])	Scale each feature to the [-1, 1] range without breaking the sparsity.
`preprocessing.minmax_scale`(X[, ...])	Transform features by scaling each feature to a given range.
`preprocessing.normalize`(X[, norm, axis, ...])	Scale input vectors individually to unit norm (vector length).
`preprocessing.quantile_transform`(X, *[, ...])	Transform features using quantiles information.
`preprocessing.robust_scale`(X, *[, axis, ...])	Standardize a dataset along any axis
`preprocessing.scale`(X, *[, axis, with_mean, ...])	Standardize a dataset along any axis.
`preprocessing.power_transform`(X[, method, ...])	Power transforms are a family of parametric, monotonic transformations that are applied to make data more Gaussian-like.

Model training support tools

- Functions that help prepare and organise datasets for training and model development:
  - Train-test split
  - Cross-validation
    - cross_val_score()
    - cross_validate()
  - Hyperparameter tuning – Grid search, Random Grid search

Hyper-parameter optimizers	
model_selection.GridSearchCV(estimator, ...)	Exhaustive search over specified parameter values for an estimator.
model_selection.HalvingGridSearchCV(... [, ...])	Search over specified parameter values with successive halving.
model_selection.ParameterGrid(param_grid)	Grid of parameters with a discrete number of values for each.
model_selection.ParameterSampler(...[, ...])	Generator on parameters sampled from given distributions.
model_selection.RandomizedSearchCV(... [, ...])	Randomized search on hyper parameters.
model_selection.HalvingRandomSearchCV(... [, ...])	Randomized search on hyper parameters.

Data modelling functions

- Supervised learning
  - Regression
    - Simple
    - Regularised regression
  - Classification
    - Binary
    - Multi-class

- Unsupervised learning
  - Clustering, k-means
  - Dimension reduction, e.g. PCA

Built-in datasets

7.1. Toy datasets

- 7.1.1. Boston house prices dataset
- 7.1.2. Iris plants dataset
- 7.1.3. Diabetes dataset
- 7.1.4. Optical recognition of handwritten digits dataset
- 7.1.5. Linnerrud dataset
- 7.1.6. Wine recognition dataset
- 7.1.7. Breast cancer wisconsin (diagnostic) dataset

7.2. Real world datasets

- 7.2.1. The Olivetti faces dataset
- 7.2.2. The 20 newsgroups text dataset
- 7.2.3. The Labeled Faces in the Wild face recognition dataset
- 7.2.4. Forest covertypes
- 7.2.5. RCV1 dataset
- 7.2.6. Kddcup 99 dataset
- 7.2.7. California Housing dataset

7.3. Generated datasets

- 7.3.1. Generators for classification and clustering
- 7.3.2. Generators for regression
- 7.3.3. Generators for manifold learning
- 7.3.4. Generators for decomposition

7.4. Loading other datasets

- 7.4.1. Sample images
- 7.4.2. Datasets in svmlight / libsvm format
- 7.4.3. Downloading datasets from the openml.org repository
- 7.4.4. Loading from external datasets

Built-in datasets


```
1  # Toy regression data set loading
2  from sklearn.datasets import load_boston
3
4  X,y = load_boston(return_X_y = True)
5
6  # Synthetic regresion data set loading
7  from sklearn.datasets import make_regression
8
9  X,y = make_regression(n_samples=10000, noise=100, random_state=0)
```

- Also, make_classification()

The pipeline command

- Useful command to streamline and data processing modelling:

```
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

X,y = load_breast_cancer(return_X_y = True)

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, random_state=0)

# Chain together scaling the variables with the model
pipe = Pipeline([('scaler', StandardScaler()), ('rf', RandomForestClassifier())])
pipe.fit(X_train, y_train)

pipe.score(X_test, y_test)
```

EXERCISE

More information and resources

- A gentle introduction to Scikit-learn: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/
- Scipy Lecture Notes - One document to learn numerics, science, and data with Python: https://scipy-lectures.org/
- Towards Data Science – A Beginner's Guide to Scikit-learn: https://towardsdatascience.com/a-beginners-guide-to-scikit-learn-14b7e51d71a4
- Just Google: 'scikit learn tutorial'


TensorFlow

What it does?

Documentation

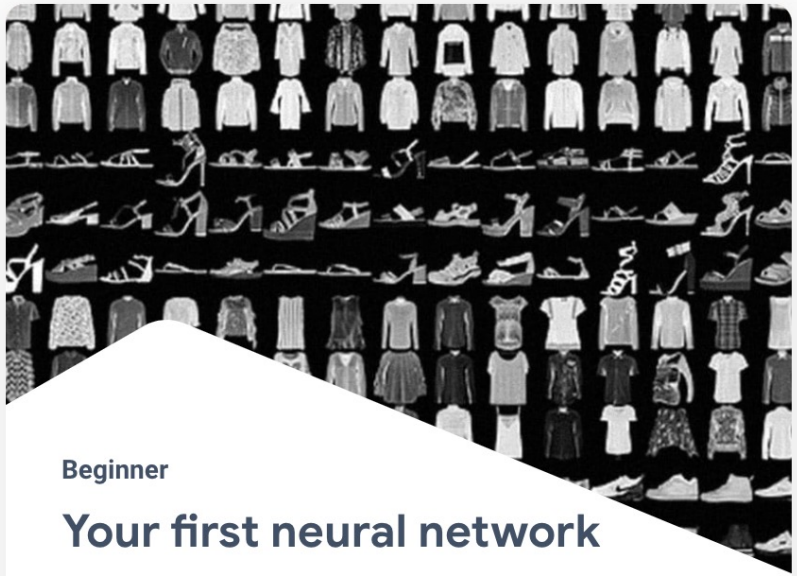
TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

What it does?

TensorFlow

Solutions to common ML problems

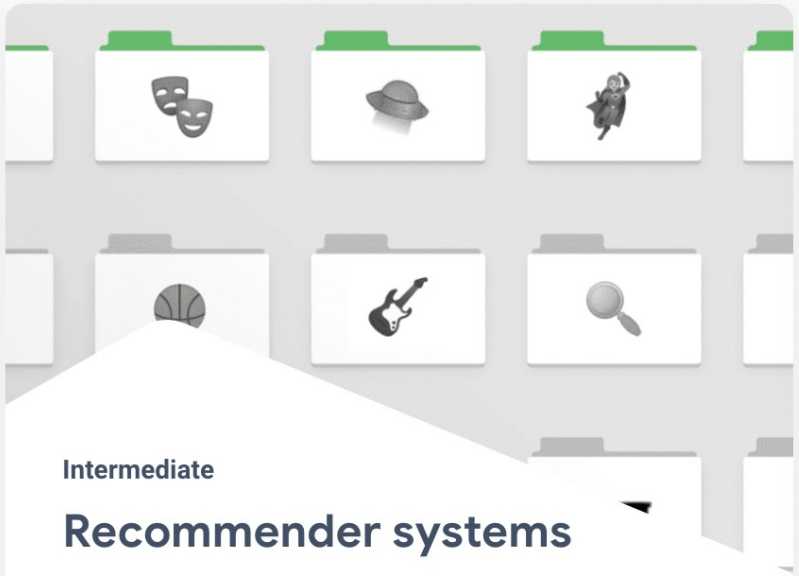
Simple step-by-step walkthroughs to solve common ML problems with TensorFlow.



Beginner

Your first neural network

Train a neural network to classify images of clothing, like sneakers and shirts, in this fast-paced overview of a complete TensorFlow program.



Intermediate

Recommender systems

Start with building and training a retrieval model to predict a set of movies that a user is likely to watch, and then use a ranking model to create recommendations.

Expert

Generative adversarial networks

Train a generative adversarial network to generate images of handwritten digits, using the Keras Subclassing API.

The TensorFlow Ecosystem

TensorFlow

TensorFlow

Learn the foundation of TensorFlow with tutorials for beginners and experts to help you create your next machine learning project.

Learn more

For JavaScript

Use TensorFlow.js to create new machine learning models and deploy existing models with JavaScript.

Learn more

For Mobile & IoT

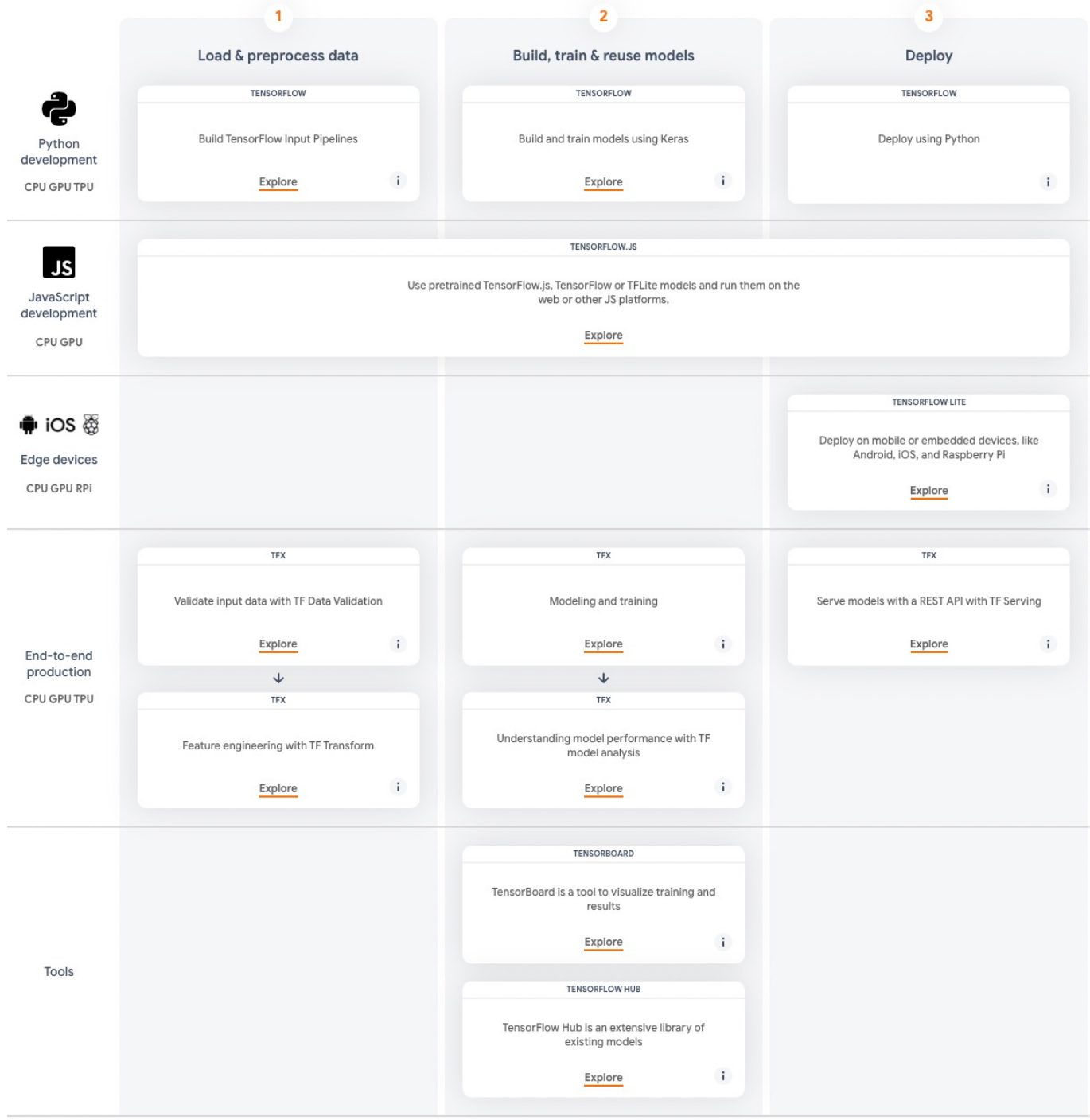
Run inference with TensorFlow Lite on mobile and embedded devices like Android, iOS, Edge TPU, and Raspberry Pi.

Learn more

For Production

Deploy a production-ready ML pipeline for training and inference using TensorFlow Extended (TFX).

Learn more


1
Load & preprocess data
2
Build, train & reuse models
3
Deploy
Python development
CPU GPU TPU
TENSORFLOW
Build TensorFlow Input Pipelines
Explore
TENSORFLOW
Build and train models using Keras
Explore
TENSORFLOW
Deploy using Python
JavaScript development
CPU GPU
TENSORFLOW.JS
Use pretrained TensorFlow.js, TensorFlow or TFLite models and run them on the web or other JS platforms.
Explore
Edge devices
CPU GPU RPi
TENSORFLOW LITE
Deploy on mobile or embedded devices, like Android, iOS, and Raspberry Pi
Explore
End-to-end production
CPU GPU TPU
TFX
Validate input data with TF Data Validation
Explore
TFX
Feature engineering with TF Transform
Explore
TFX
Modeling and training
Explore
TFX
Understanding model performance with TF model analysis
Explore
TFX
Serve models with a REST API with TF Serving
Explore
Tools
TENSORBOARD
TensorBoard is a tool to visualize training and results
Explore
TENSORFLOW HUB
TensorFlow Hub is an extensive library of existing models
Explore

The TensorFlow Ecosystem

TensorFlow

Python development

CPU GPU TPU

1

Load & preprocess data

TENSORFLOW

Build TensorFlow Input Pipelines

Explore

2

Build, train & reuse models

TENSORFLOW

Build and train models using Keras

Explore

3

Deploy

TENSORFLOW

Deploy using Python

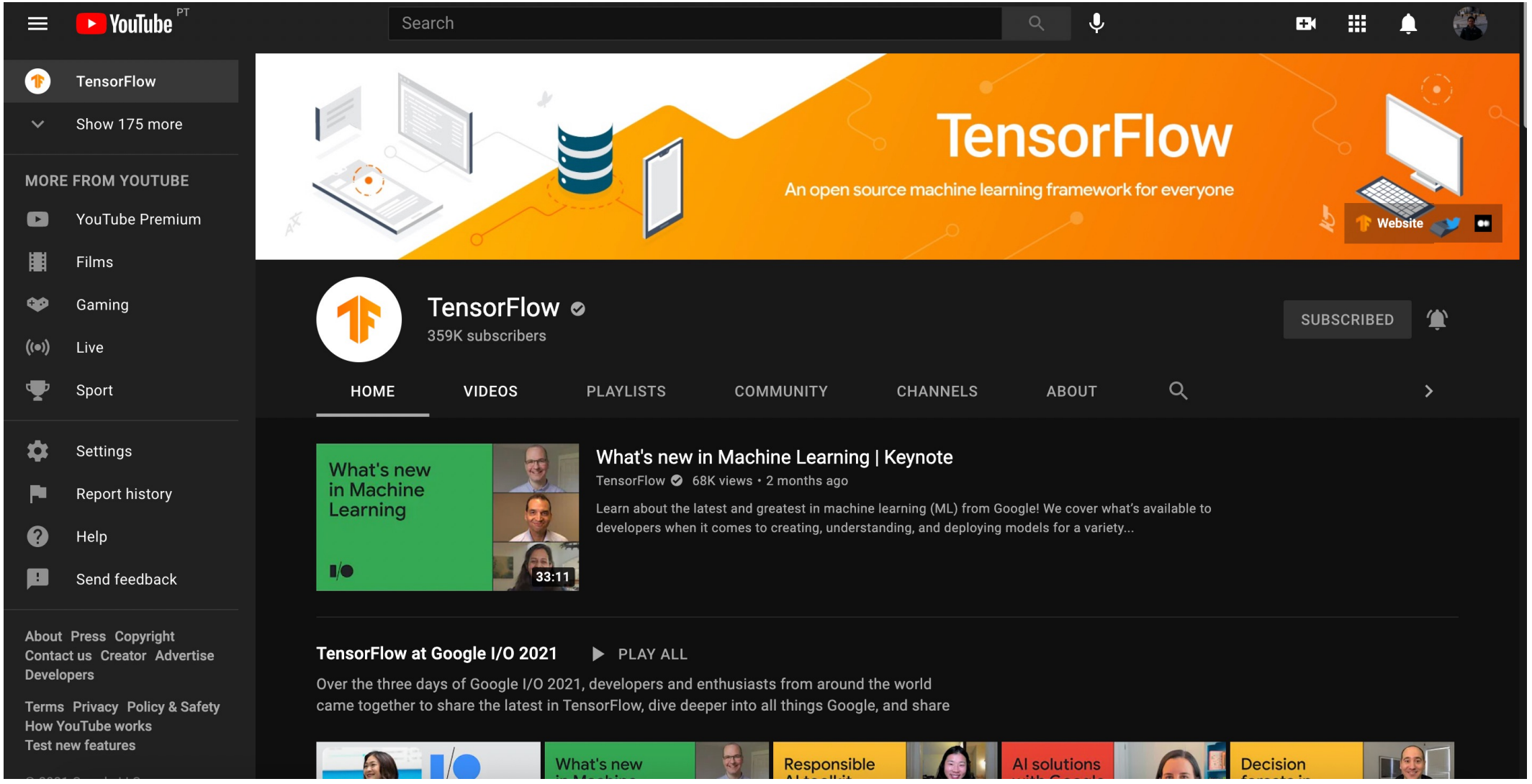
Methods implemented

- Implementation of statistical models based on tensor data structures
  - Regression
  - Classification
  - Deep Learning
    - Neural Networks
    - Convolutional Neural Networks
    - Sequence models
    - General Adversarial Networks
    - Reinforcement Learning

EXERCISE

More information and resources

TensorFlow



More information and resources

- TensorFlow 2.0 Complete Course - Python Neural Networks for Beginners Tutorial in 7 hours: https://www.youtube.com/watch?v=tPYj3fFJGjk

  (https://www.freecodecamp.org/)

- DeepLearning.AI TensorFlow Developer Professional Certificate: https://www.coursera.org/professional-certificates/tensorflow-in-practice

- Tutorials Point: https://www.tutorialspoint.com/tensorflow/index.htm


PyTorch
geometric

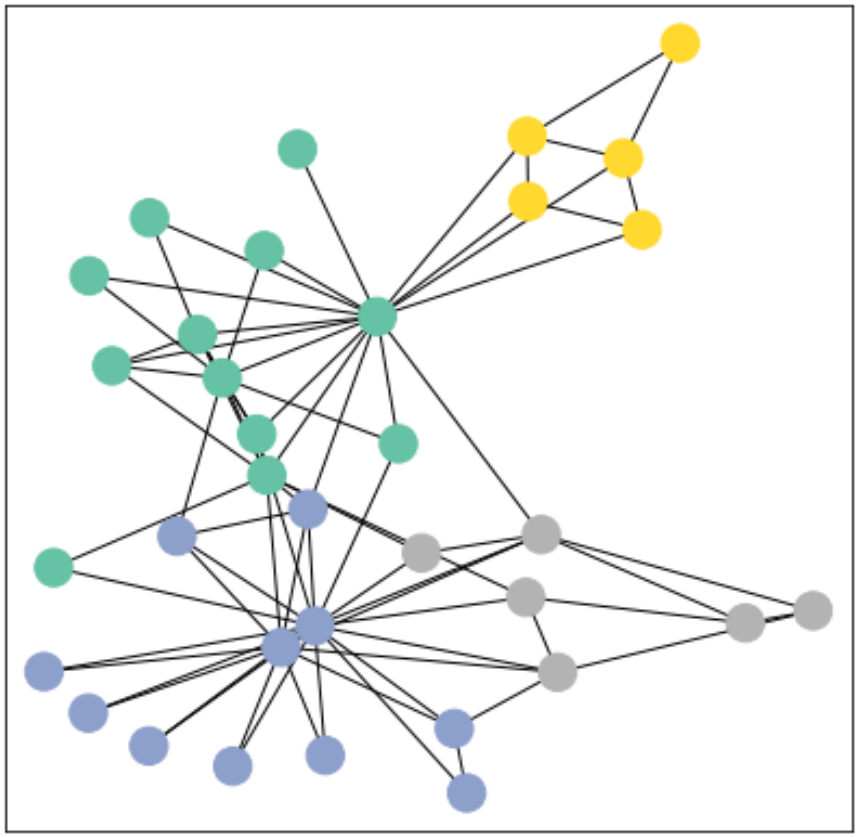
What it does?

- Extension of Pytorch for geometric deep learning
- Allows analysis of graph structured data using Graph Neural Networks (GNN)
- Includes uniform implementation of over 40 GNN operators/models (with frequent updates)
- 100 benchmark datasets
- Automatic mini-batching, deterministic and differentiable pooling operators.
- Data transformation, like augmentation and point sampling

Graph-structured data

PyTorch geometric

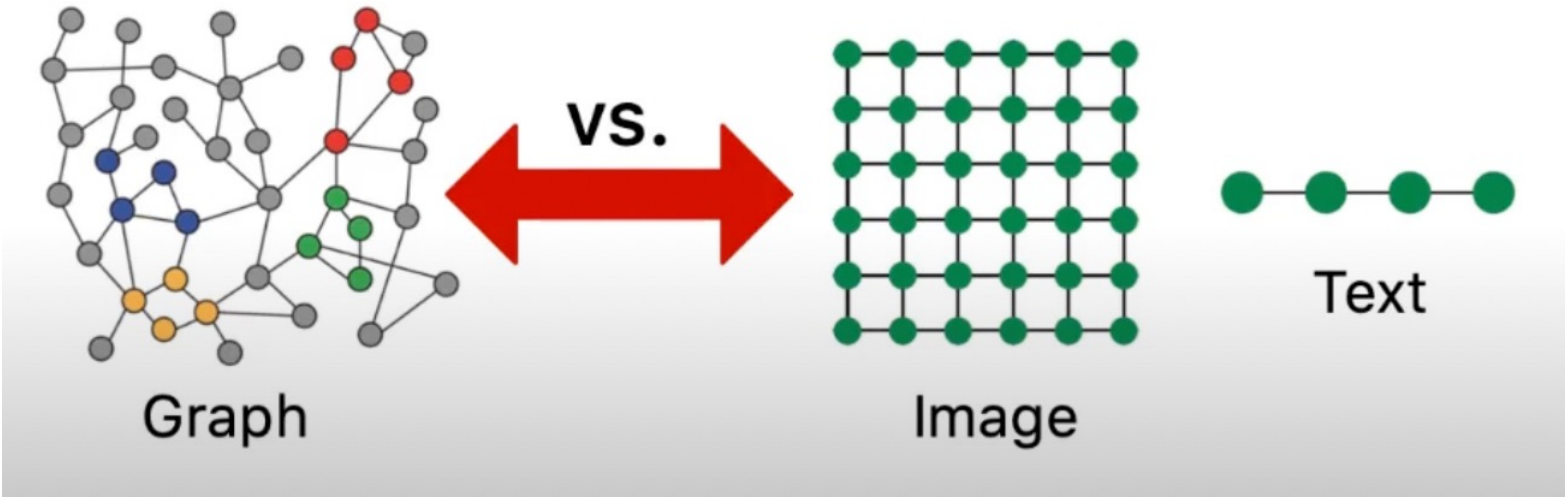
- Nodes
- Edges
- Attributes/Features
  - Node attributes
  - Edge attributes



Graph-structured data

PyTorch geometric

- Graph data can be seen as a generalisation of other types of structured data



Graph-structured data

- Graph data can be seen as a generalisation of other types of structured data

- Captures more complex relationships and (disorganized) structures
  - Biological networks
  - Social networks
  - Communication networks
  - Knowledge graphs

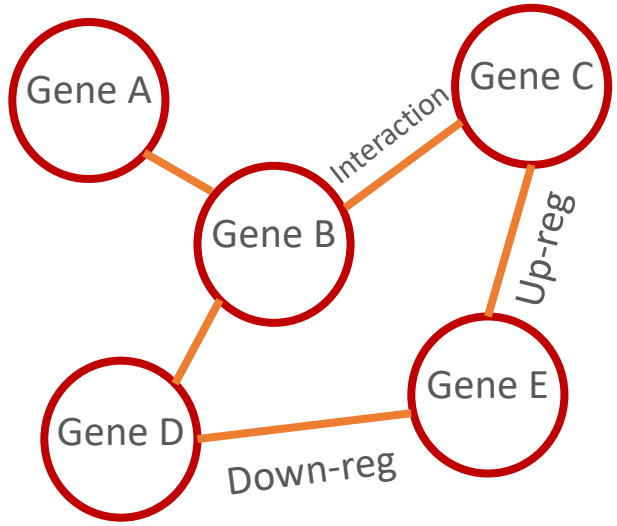
Learning on graphs

PyTorch geometric

1. Graph classification – graph level
   - Provides a discrete classification given the nodes and edges of a graph

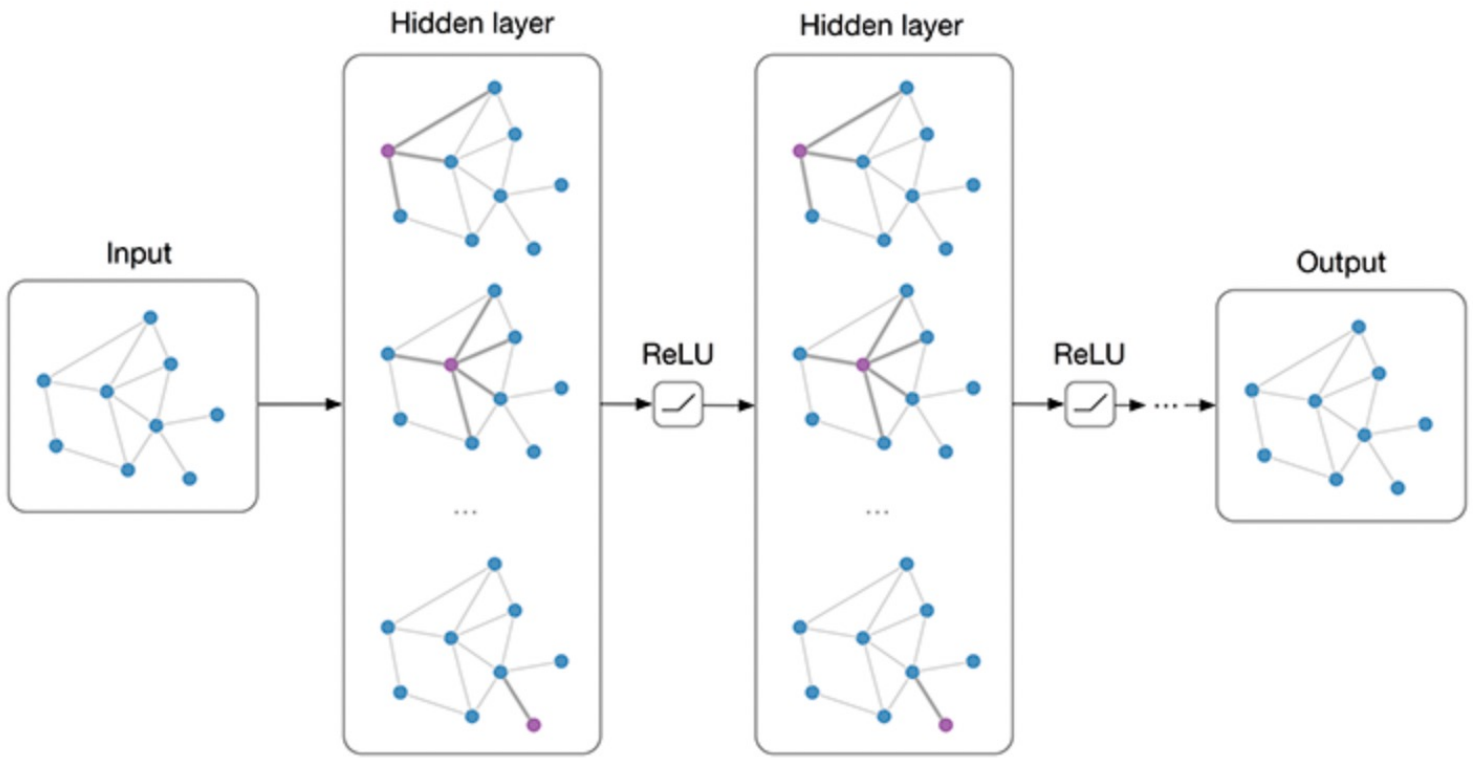
2. Node classification - node-level
   - Attributes a classification to the nodes of a graph

3. Link prediction – edge-level
   - Learn from the graph structure and predicts novel links



EXERCISE

THOMAS KIPF, 30 SEPTEMBER 2016



Multi-layer Graph Convolutional Network (GCN) with first-order filters.

EXTERNAL RESOURCES

- Matthias Fey and Jan E. Lenssen: **Fast Graph Representation Learning with PyTorch Geometric** [Paper, Slides (3.3MB), Poster (2.3MB), Notebook]
- Soumith Chintala: **Automatic Differentiation, PyTorch and Graph Neural Networks** [Talk (starting from 26:15)]
- Steeve Huang: **Hands-on Graph Neural Networks with PyTorch & PyTorch Geometric** [Tutorial, Code]
- Stanford University: **Graph Neural Networks using PyTorch Geometric** [Talk (starting from 33:33)]
- Francesco Landolfi: **PyTorch Geometric Tutorial** [PDF (0.4MB)]
- Nicolas Chaulet *et al.*: **PyTorch Points 3D** - A framework for running common deep learning models for point cloud analysis tasks that heavily relies on Pytorch Geometric [Github, Documentation]
- Weihua Hu *et al.*: **Open Graph Benchmark** - A collection of large-scale benchmark datasets, data loaders, and evaluators for graph machine learning, including PyTorch Geometric support and examples [Website, GitHub]
- **DeepSNAP** - A PyTorch library that bridges between graph libraries such as NetworkX and PyTorch Geometric [GitHub, Documentation]
- Benedek Rozemberczki: **PyTorch Geometric Temporal** - A temporal GNN library built upon PyTorch Geometric [GitHub, Documentation]
- Amitoz Azad: **torch_pdegraph** - Solving PDEs on graphs with PyTorch Geometric [Devpost, GitHub]
- Antonio Longa, Gabriele Santin and Giovanni Pellegrini: **PyTorch Geometric Tutorial** [Website, GitHub]
- Amitoz Azad: **Primal-Dual Algorithm for Total Variation Processing on Graphs** [Jupyter]

Honorable mentions

1. Siuba – dplyr functions in Python for data management and processing https://github.com/machow/siuba

scrappy data analysis, with seamless support for pandas and SQL

CI passing | docs passing | launch binder

siuba (小巴) is a port of dplyr and other R libraries. It supports a tabular data analysis workflow centered on 5 common actions:

- `select()` - keep certain columns of data.
- `filter()` - keep certain rows of data.
- `mutate()` - create or modify an existing column of data.
- `summarize()` - reduce one or more columns down to a single number.
- `arrange()` - reorder the rows of data.

These actions can be preceeded by a `group_by()`, which causes them to be applied individually to grouped rows of data. Moreover, many SQL concepts, such as `distinct()`, `count()`, and joins are implemented. Inputs to these functions can be a pandas `DataFrame` or SQL connection (currently postgres, redshift, or sqlite).

For more on the rationale behind tools like dplyr, see this tidyverse paper. For examples of siuba in action, see the siuba documentation.

Honorable mentions

1. Siuba – dplyr functions in Python for data management and processing https://github.com/machow/siuba

```
from siuba import group_by, summarize, _
from siuba.data import mtcars

(mtcars
  >> group_by(_.cyl)
  >> summarize(avg_hp = _.hp.mean())
  )
```

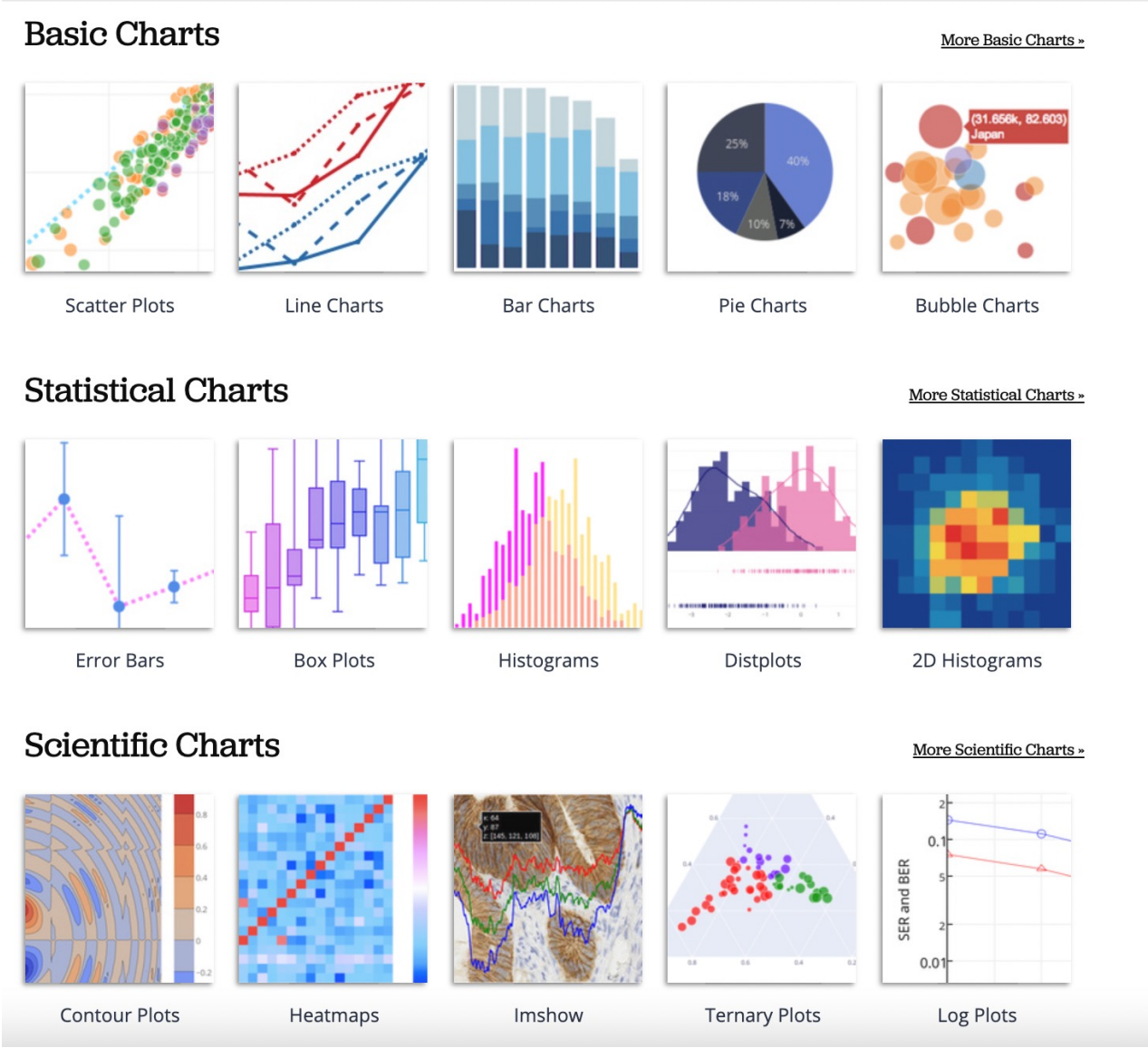
```
Out[1]:
   cyl       avg_hp
0    4    82.636364
1    6   122.285714
2    8   209.214286
```

Honorable mentions

2. Plotly https://plotly.com/python/
   - Open-source plotting library built on top of Plotly JavaScript
   - Let’s users create web-based visualisations
   - Support over 40 different chart types with multiple applications
   - **Interactive!**
   - Jupyter Notebook and Jupyter Lab support
   - Static Image export utilities
     - PyCharm
     - Static documents like PDFs

Honorable mentions

2. Plotly https://plotly.com/python/



Honorable mentions

- Dash https://github.com/plotly/dash

Dash is the most downloaded, trusted Python framework for building ML & data science web apps.

Built on top of Plotly.js, React and Flask, Dash ties modern UI elements like dropdowns, sliders, and graphs directly to your analytical Python code. Read our tutorial proudly crafted ❤️ by Dash itself.

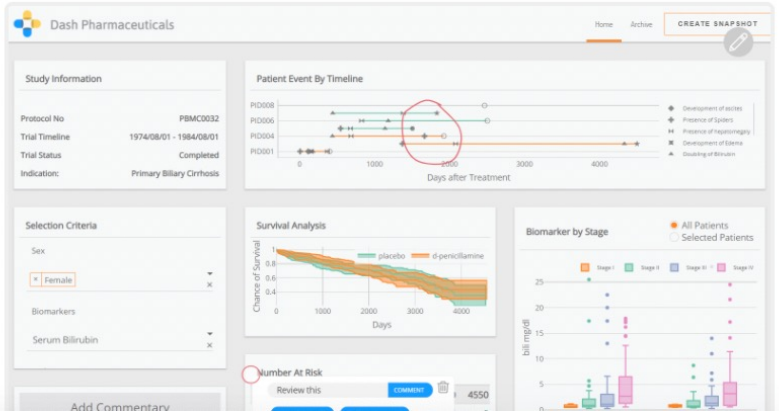
Dash Enterprise App Gallery

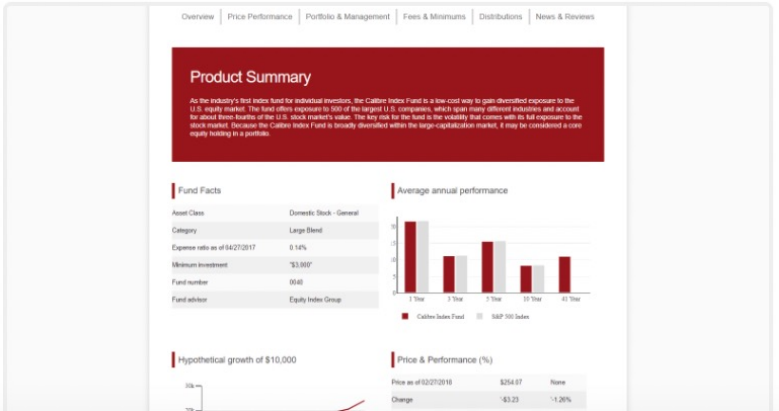
This public instance of the 👑 Dash Enterprise 👑 app manager runs >60 Dash apps for 100s of concurrent users on Azure Kubernetes Service. Click on a Dash app's name to below for more information. For the open-source demos, the Python & R source code can be found on GitHub. For apps using Design Kit or Snapshot Engine, reach out to get a demo.

Aerospace | Automotive | Energy | Finance | Manufacturing | Medical Imaging | Pharma | Retail | Sports Analytics

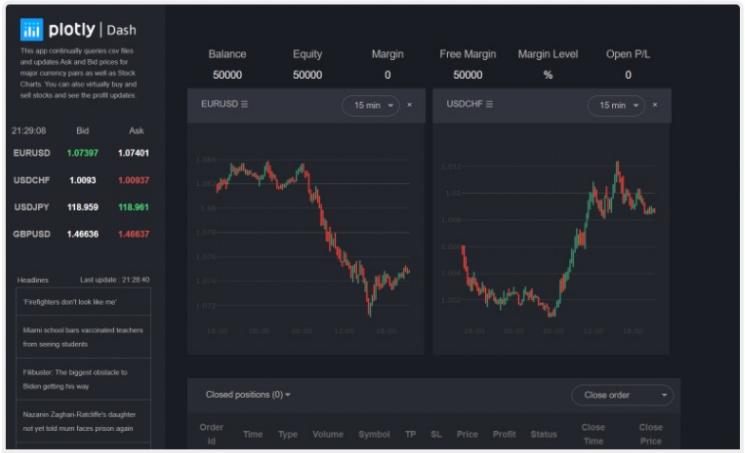
All Apps (117)

Search applications...



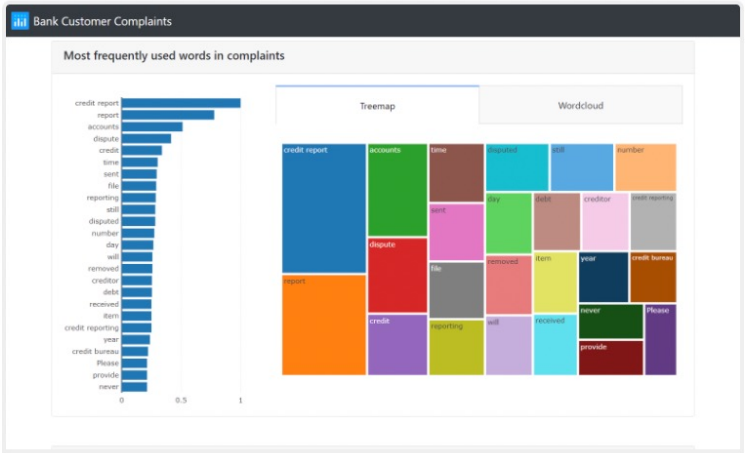


plotly | Dash Enterprise



FOREX Web Trader

Streaming | Financial

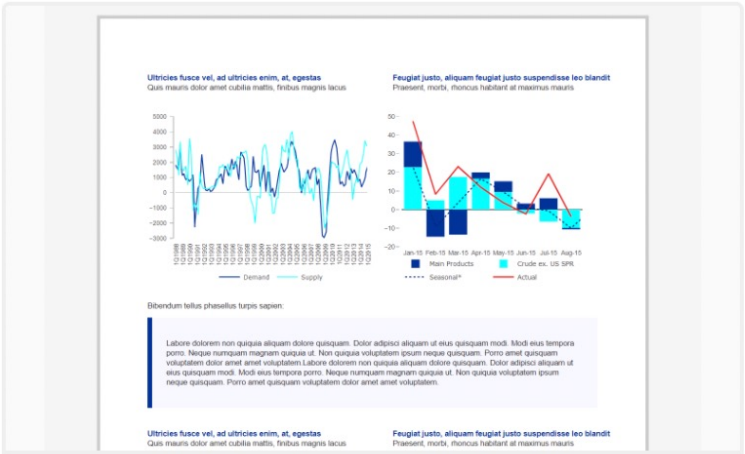


Bank Complaints NLP

AI | ML | NLP | Financial | Insurance | Report | (+1)

t-SNE Explorer

AI | ML | Computer Vision | scikit-learn | Financial

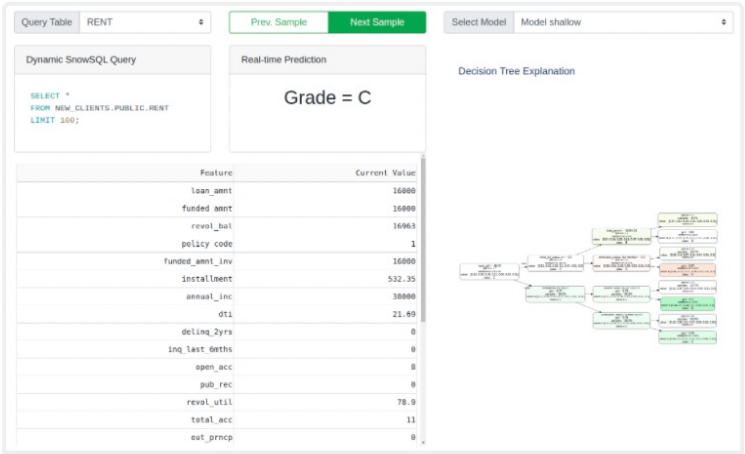


Multipage Report

Financial | Report

Yield Curve

Financial | 3d | Insurance



Loan Grade Classification AI

xAI | AI | ML | scikit-learn | Snowflake | (+3)

Be in touch!

miguel@plaindata.ai | https://plaindata.ai/