Affective User Research & Human-AI Interaction

Seminar Summer 2024, Karlsruhe Institute of Technology

Dr. Ivo Benke, BioNTech
Dr. Lennard Schmidt, Google

Karlsruhe Institute of Technology

human-centered systems lab

This image was created with the assistance of DALL·E 2

Karlsruhe Institute of Technology

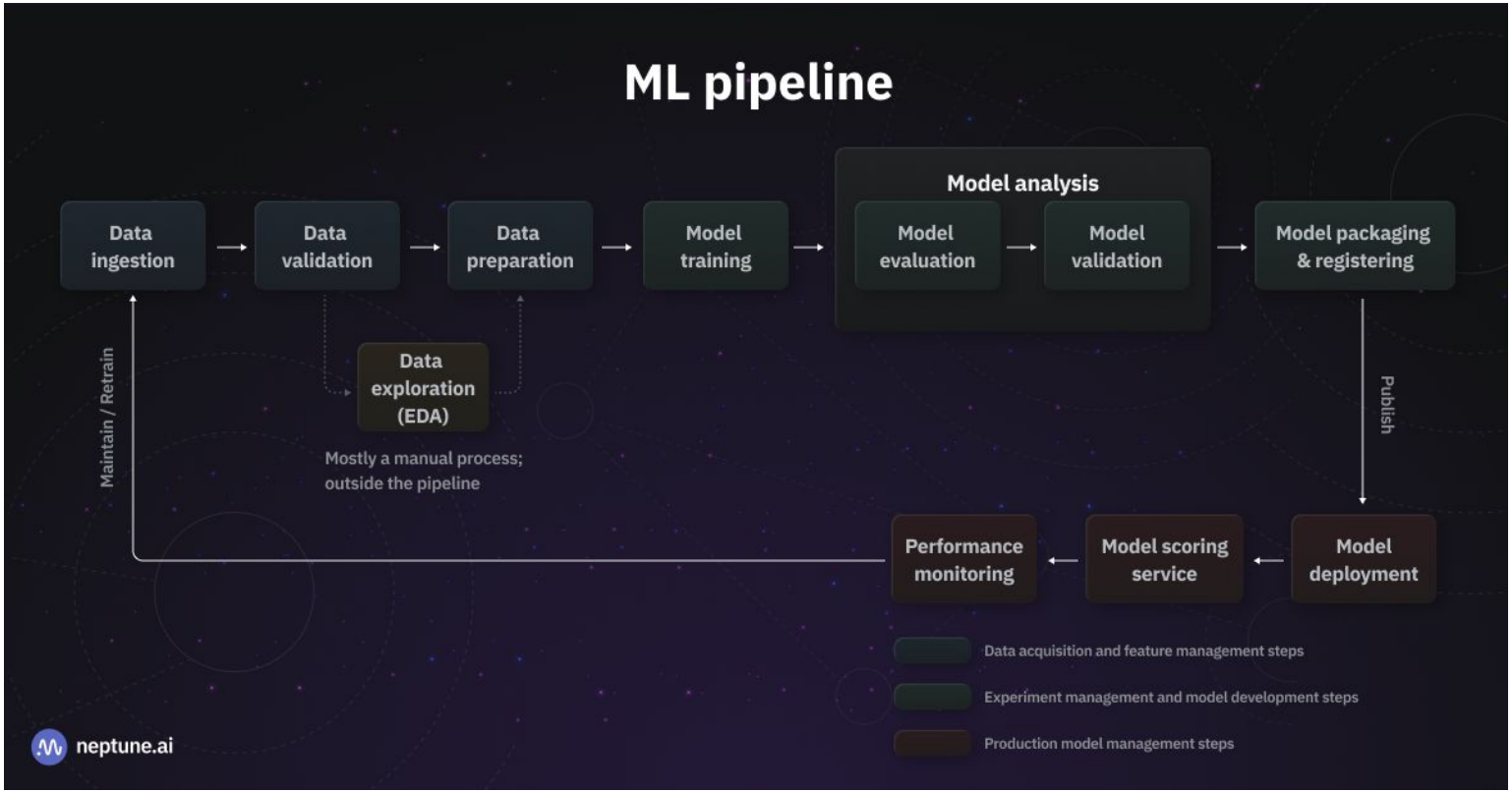
human-centered systems lab

Affective User Research & Human-AI Interaction

Seminar #6 Part: Data Science Methods

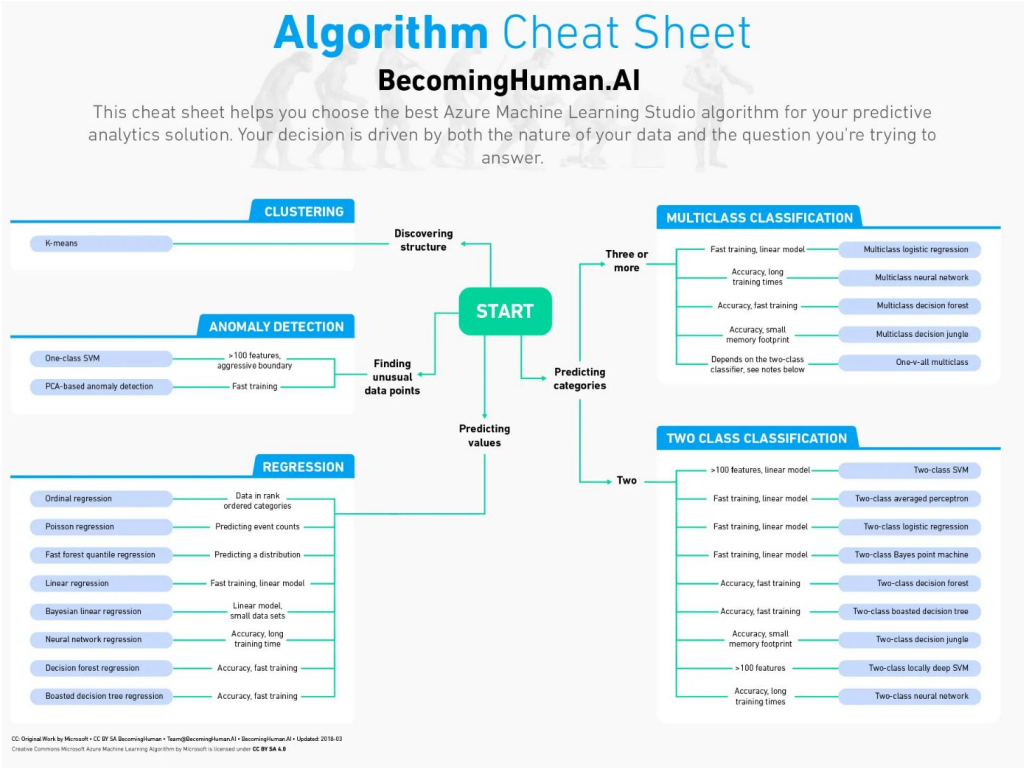
Dr. Ivo Benke, Dr. Lennard Schmidt

ML Pipeline


ML pipeline
Data ingestion
Data validation
Data preparation
Model training
Model analysis
Model evaluation
Model validation
Model packaging & registering
Data exploration (EDA)
Mostly a manual process; outside the pipeline
Maintain / Retrain
Publish
Performance monitoring
Model scoring service
Model deployment
Data acquisition and feature management steps
Experiment management and model development steps
Production model management steps
neptune.ai

Machine Learning
Helpful Packages

Model Refresher



Scikit-Learn

- A popular Python library for machine learning that provides simple and efficient tools for data mining and data analysis.
- Built on NumPy, SciPy, and matplotlib.
- Supports supervised and unsupervised learning with a rich documentation and active community.

- Main sections in User Guide
  - 1. Supervised learning
  - 2. Unsupervised learning
  - 3. Model selection and evaluation
  - 5. Visualizations
  - 6. Dataset transformations
  - 10. Common pitfalls and recommended practices

Scikit Learn Cheat Sheet

Cheat-Sheet Skicit learn
Phyton For Data Science
BecomingHuman.AI DataCamp

Skicit Learn
Skicit Learn is an open source Phyton library that implements a range if machine learning, processing, cross validation and visualization algorithm using a unified

A basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris() >>> X, y = iris.data[:, :2], iris.target
>>> Xtrain, X test, y_train, y test = train_test_split (X, y, random stat33)
>>> scaler = preprocessing.StandardScaler() fit(X_train)
>>> X train = scaler.transform(X train)
>>> X test = scaler.transform(X test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_scorely_test, y_pred)
```

Prediction
Supervised Estimators — Predict labels; Predict labels; Estimate probability of a label
```
>>> y_pred = svc.predict(np.random.random(2,5))
>>> y_pred = lr.predict(X_test)
>>> y_pred = knn.predict_proba(X_test)
```
Unsupervised Estimators — Predict labels in clustering algos
```
>>> y_pred = k_means.predict(X_test)
```

Loading the Data
Your data beeds to be nmueric and stored as NumPy arrays or SciPy sparse matric. other types that they are convertible to numeric arrays, such as Pandas Dataframe, are also acceptable

```
>>> import numpy as np >> X = np.random.random((10,5))
>>> y = np . array ( ['M', 'M', 'F', 'F' , 'M', 'F', 'M', 'M', 'F', 'F', 'F' ])
>>> X [X < 0.7] = 0
```

Preprocessing The Data
Standardization
```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler() fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```
Normalization
```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer() fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```
Binarization
```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0) fit(X)
>>> binary_X = binarizer.transform(X)
```
Encoding Categorical Features
```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```
Imputing Missing Values
```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy= mean, axis=0)
>>> imp.fit_transformX_train)
```
Generating Polynomial Features
```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

Evaluate Your Model's Performance
Classification Metrics

Accuracy Score — Estimator score method; Metric scoring functions
```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import accuracy_score
>>> accuracy_scorely_test, y_pred)
```
Classification Report — Precision, recall, f1-score and support
```
>>> from sklearn.metrics import classification_report
>>> printclassification_reportly_test, y_pred))
```
Confusion Matrix
```
>>> from sklearn.metrics import confusion_matrix
>>> printconfusion_matrixly_test, y_pred)
```

Regression Metrics

Mean Absolute Error
```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_errorly_true, y_pred)
```
Mean Squared Error
```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_errorly_test, y_pred)
```
R^2 Score
```
>>> from sklearn.metrics import r2_score
>>> r2_scorely_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index
```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_scorely_true, y_pred)
```
Homogeneity
```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_scorely_true, y_pred)
```
V-measure
```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_scorely_true, y_pred)
```

Cross-Validation
```
>>> from sklearn.cross_validation import cross_val_score
>>> printcross_val_score(knn, X_train, y_train, cv=4))
>>> printcross_val_score(lr, X, y, cv=2))
```

Model Fitting
Supervised learning — Fit the model to the data
```
>>> lr.fit(X, y)
>>> knn.fit(X_train, y_train)
>>> svc.fit(X_train, y_train)
```
Unsupervised Learning — Fit the model to the data; Fit to data, then transform it
```
>>> k_means.fit(X_train)
>>> pca_model= pca.fit_transform(X_train)
```

Create Your Model
Supervised Learning Estimators

Linear Regression
```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```
Support Vector Machines (SVM)
```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```
Naive Bayes
```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```
KNN
```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)
```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```
K Means
```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

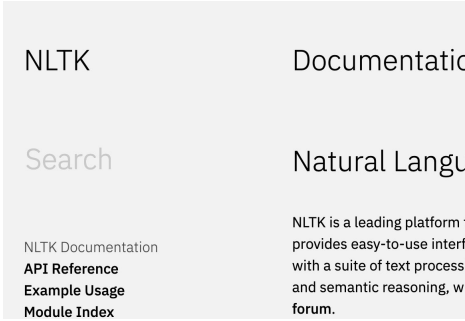
Training And Test Data
```
>> from sklearn.cross validation import train_test_split
>> X train, X test, y train, y test = train_test_split(X,
                                                       y,
                                                       random state-0)
```

Tune Your Model
Grid Search
```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3)
              "metric": ["euclidean","cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```
Randomized Parameter Optimization
```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

NLTK

What is NLTK?

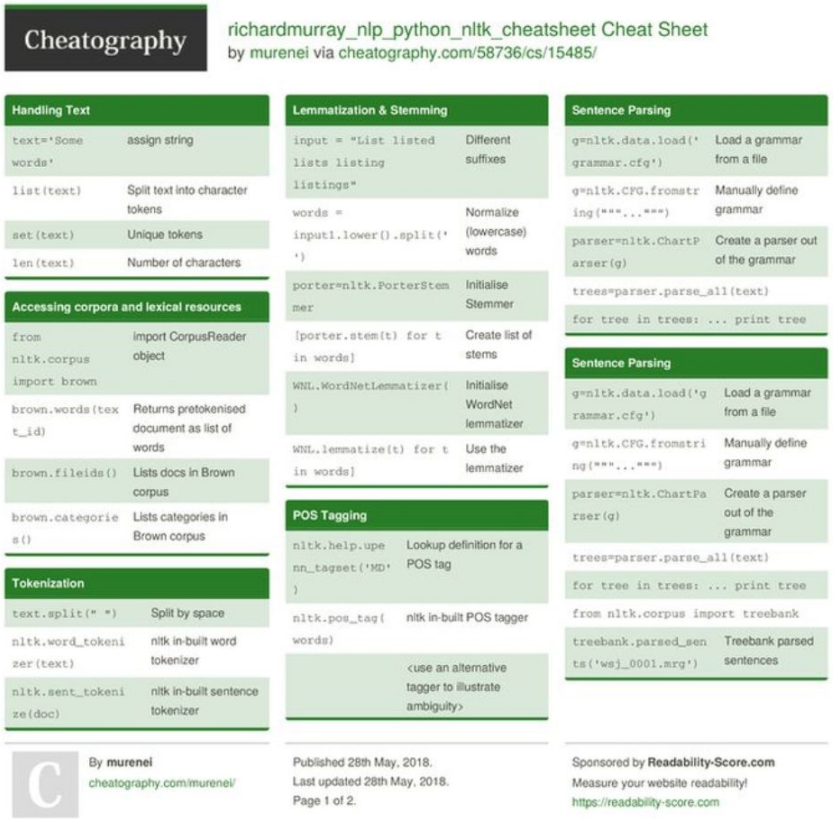
- A powerful Python library for working with human language data (text).
- Provides easy-to-use interfaces to over 50 corpora and lexical resources.
- Contains a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.



Features:

- **Corpora and Lexical Resources:** Access to large text corpora, such as Gutenberg, Brown, and many others.
- **Text Processing:** Tools for tokenizing text, finding n-grams, part-of-speech tagging, and named entity recognition.
- **Classification:** Implements various machine learning algorithms for text classification.
- **Text Analysis:** Tools for language modeling, text generation, and sentiment analysis.
- **Educational:** Comes with comprehensive documentation and tutorials for learning.

NLTK Cheat Sheet



Cheatography

richardmurray_nlp_python_nltk_cheatsheet Cheat Sheet
by murenei via cheatography.com/58736/cs/15485/

Handling Text

`text='Some words'`	assign string
`list(text)`	Split text into character tokens
`set(text)`	Unique tokens
`len(text)`	Number of characters

Accessing corpora and lexical resources

`from nltk.corpus import brown`	import CorpusReader object
`brown.words(text_id)`	Returns pretokenised document as list of words
`brown.fileids()`	Lists docs in Brown corpus
`brown.categories()`	Lists categories in Brown corpus

Tokenization

`text.split(" ")`	Split by space
`nltk.word_tokenizer(text)`	nltk in-built word tokenizer
`nltk.sent_tokenize(doc)`	nltk in-built sentence tokenizer

Lemmatization & Stemming

`input = "List listed lists listing listings"`	Different suffixes
`words = input1.lower().split(' ')`	Normalize (lowercase) words
`porter=nltk.PorterStemmer`	Initialise Stemmer
`[porter.stem(t) for t in words]`	Create list of stems
`WNL.WordNetLemmatizer()`	Initialise WordNet lemmatizer
`WNL.lemmatize(t) for t in words]`	Use the lemmatizer

POS Tagging

`nltk.help.upenn_tagset('MD')`	Lookup definition for a POS tag
`nltk.pos_tag(words)`	nltk in-built POS tagger
	<use an alternative tagger to illustrate ambiguity>

Sentence Parsing

`g=nltk.data.load('grammar.cfg')`	Load a grammar from a file
`g=nltk.CFG.fromstring("""...""")`	Manually define grammar
`parser=nltk.ChartParser(g)`	Create a parser out of the grammar
`trees=parser.parse_all(text)`	
`for tree in trees: ... print tree`	

Sentence Parsing

`g=nltk.data.load('grammar.cfg')`	Load a grammar from a file
`g=nltk.CFG.fromstring("""...""")`	Manually define grammar
`parser=nltk.ChartParser(g)`	Create a parser out of the grammar
`trees=parser.parse_all(text)`	
`for tree in trees: ... print tree`	
`from nltk.corpus import treebank`	
`treebank.parsed_sents('wsj_0001.mrg')`	Treebank parsed sentences

By **murenei**
cheatography.com/murenei/

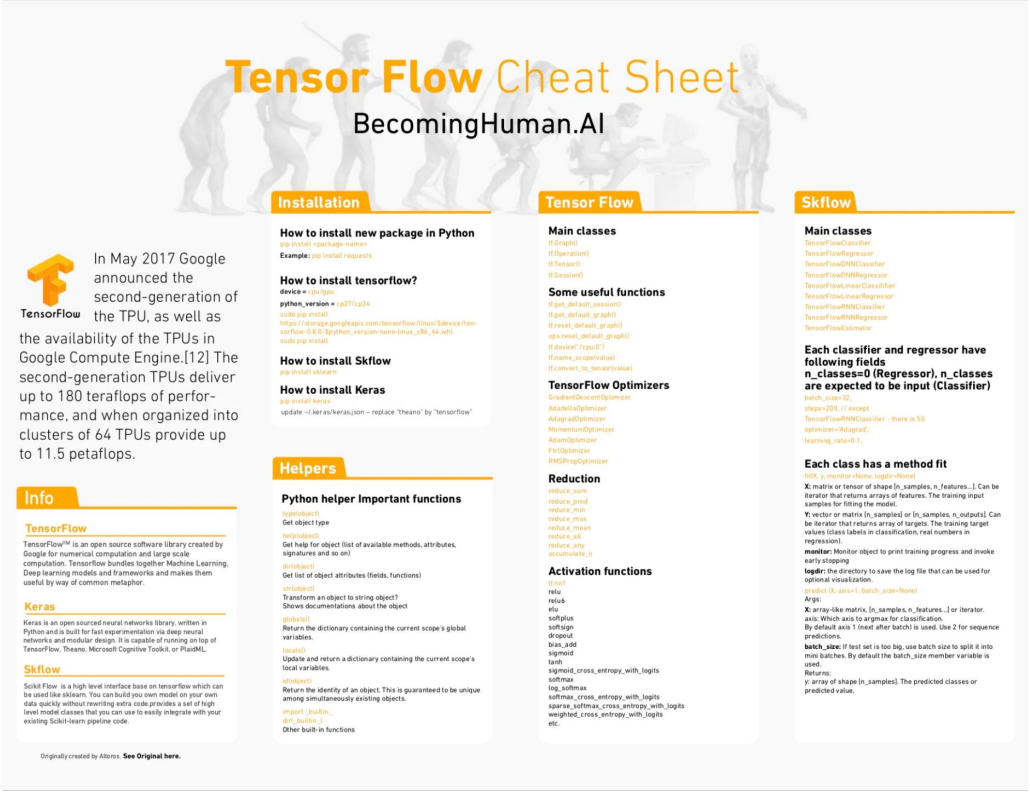
Published 28th May, 2018.
Last updated 28th May, 2018.
Page 1 of 2.

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com

Tensorflow

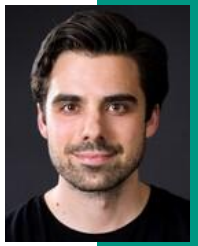
- **TensorFlow (www.tensorflow.org/api) a set of tools and functions that to build and deploy machine learning models.** Allows for different levels of abstraction, from low-level operations on tensors to high-level APIs like Keras, which simplifies the process of defining and training neural networks.
- **TensorFlow Hub (www.tensorflow.org/hub) is a repository of trained machine learning models** ready for fine-tuning and deployable anywhere. Reuse trained models like BERT and Faster R-CNN with just a few lines of code.

Tensorflow Cheat Sheet



Questions, Comments, Observations

Thank You



Dr. Ivo Benke

ivolucasbenke@gmail.com

Dr. Lennard Schmidt

lennard.schmidt@hhl.de