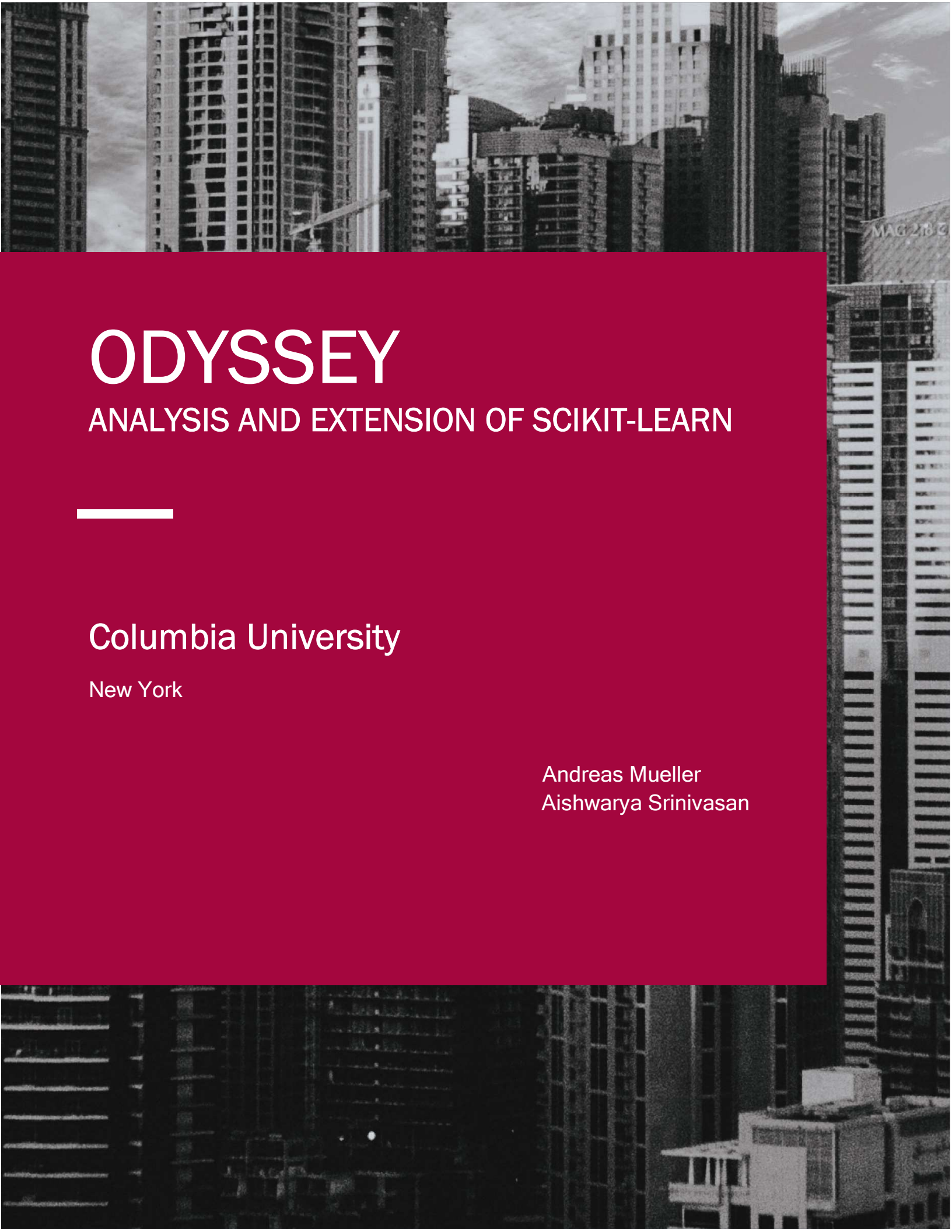


ODYSSEY

ANALYSIS AND EXTENSION OF SCIKIT-LEARN

Columbia University

New York

Andreas Mueller
Aishwarya Srinivasan

Problem Description

Scikit-learn is a Python machine learning library containing a large collection of machine learning models, as well as evaluation metrics and tools for implementing machine learning workflows.

The goal of this project is to analyze the current usage of scikit-learn on a large scale (i.e. the scale of all open-source code, even all public code), and extend the library based on the findings. We want to identify usage patterns, problematic use cases, and ways to improve the interface.

Data Source

- The main source of data is the database of all open-source Github repositories provided by Google BigQuery.
- SQL Queries were written in Python to extract data from the Google BigQuery API.
- Selective queries were written to extract data related to specific packages.

- We will build on the Odyssey library which was developed to support this project. (https://github.com/alan97/odyssey)

Things to explore

- Compare different modules, functions, classes between ipynb and python files
  Scatter plot for number of imports of sklearn modules in ipynb and py files
  Odyssey_compare_python_ipynb file contains the analysis of the sklearn, matplotlib numpy

- Usages of Pipeline
  Odyssey_pipeline_analysis contains the analysis of the pipeline usages in python and ipython files.

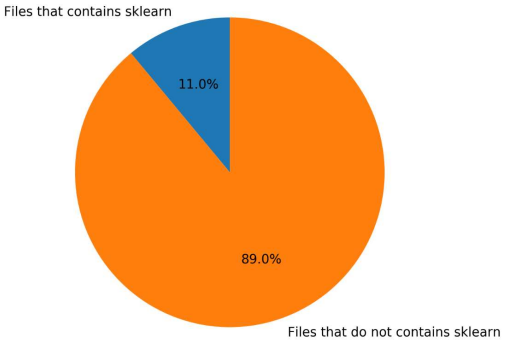
- Information of all function/module imports using jedi - in py and ipynb files

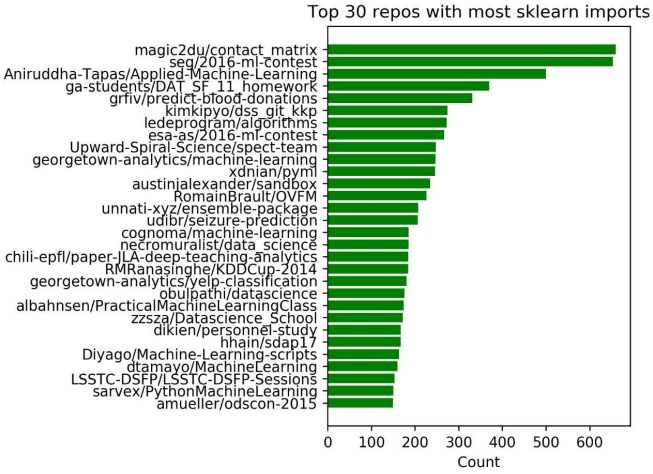
- Hyperparameters of functions
  Using nb convert - convert the json files of ipynb to py files

- Instantiation Analyser/ jedi on both Pipeline and MakePipeline.

Scikit-Learn Analysis

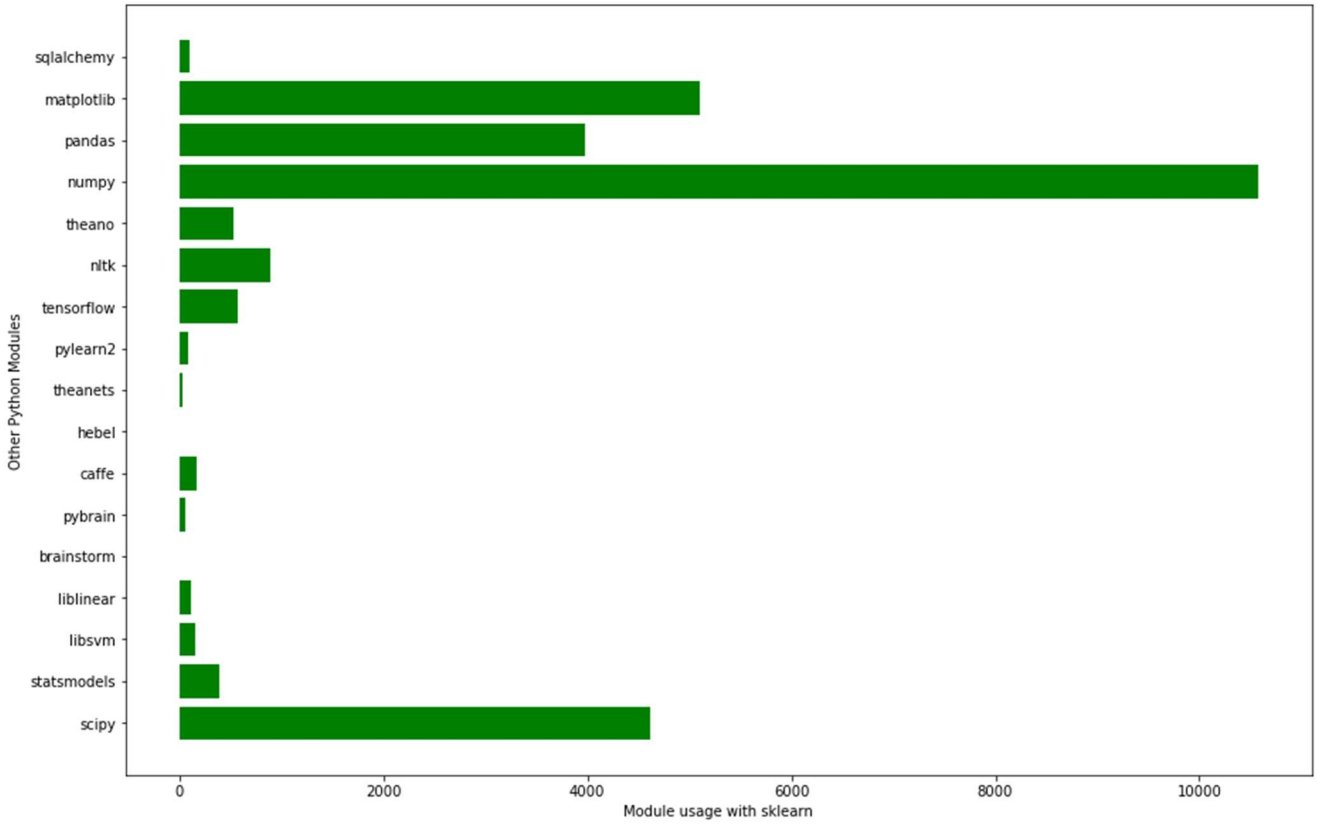
- Initial analysis includes understanding the usage of sklearn and understanding the repos with most imports of the sklearn package.



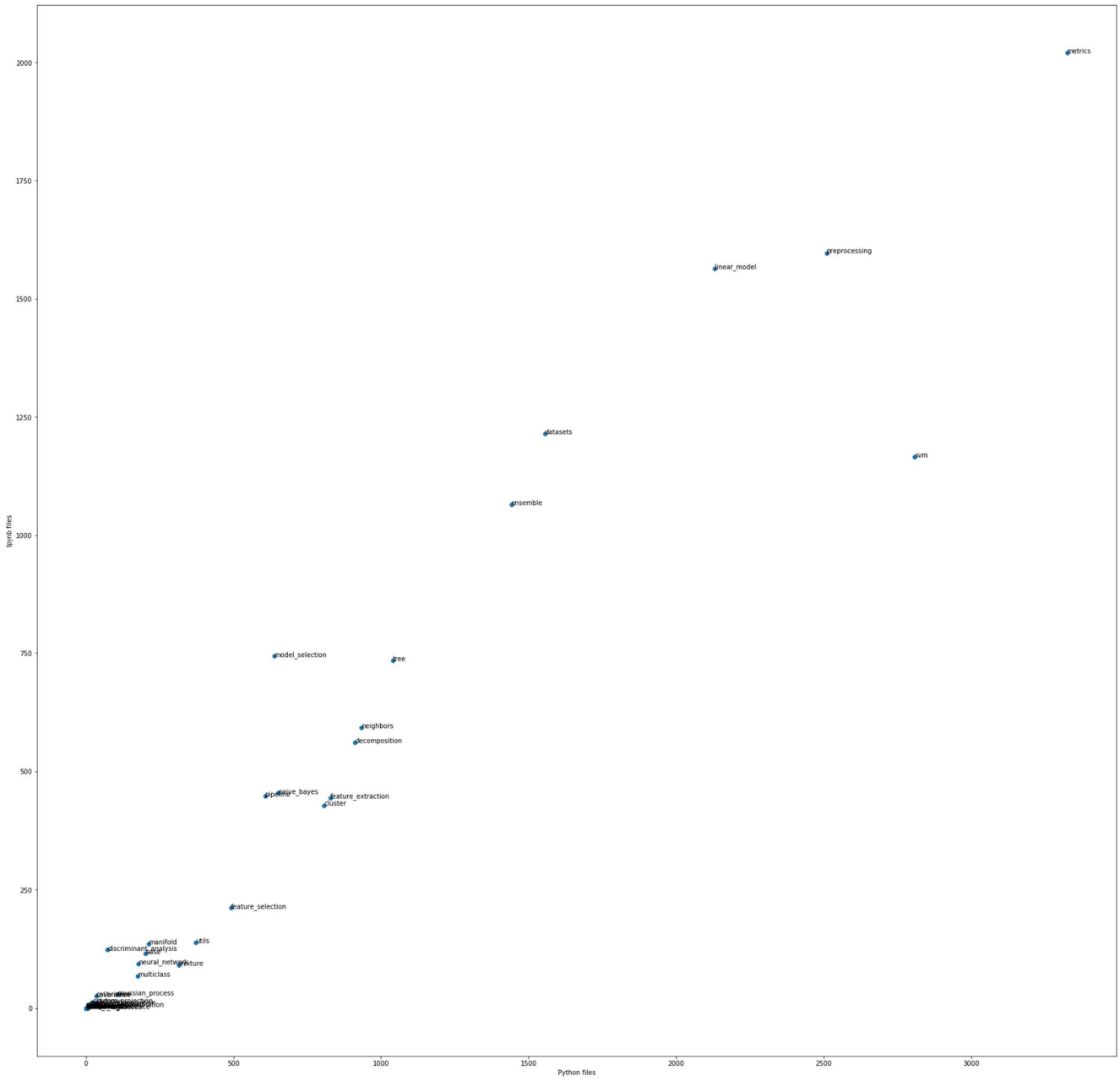


- To have a comparative analysis of the Scikit-learn usage with other modules of Python. For start we will be looking into the following modules:

['sqlalchemy','matplotlib','pandas', 'numpy', 'theano', 'nltk', 'tensorflow', 'pylearn2', 'theanets', 'hebel', 'caffe', 'pybrain', 'brainstorm', 'liblinear', 'libsvm', 'statsmodels', 'scipy']


sqlalchemy
matplotlib
pandas
numpy
theano
nltk
tensorflow
pylearn2
theanets
hebel
caffe
pybrain
brainstorm
liblinear
libsvm
statsmodels
scipy
Other Python Modules
0
2000
4000
6000
8000
10000
Module usage with sklearn

- Looking into the usages of the various sub modules of sklearn in ipynb and python files



- Hyperparameter usages for all the functions of Sklearn (both ipynb and python files). Use instantiation analyzer to parse the function usages.

  The analysis is done in the “Odyssey_sklearn_hyperparameter.ipynb”

  It is required to access the hyperparameters by reading the pickle file and using the key of the function name.

  For example, hyperparameter_ipynb[‘RandomForestClassifier’][‘max_depth’]

- Usage of the “pipeline” function of sklearn, to understand the streamline of the function being implemented.

  Jedi python library is used to parse the function usages, parameter declaration and trace back to the package imports of the functions from their libraries.

  Analysis could be found here: “Odyssey_pipeline_analysis.ipynb”

	level_0	Description	Package/Module	Usage	index
0	3	class Pipeline	sklearn.pipeline.Pipeline	from sklearn.pipeline import Pipeline	231
1	12	instance Pipeline	sklearn.pipeline.Pipeline	clf = Pipeline(steps=[('rbm', rbm), ('logistic...	231
2	13	instance Pipeline	sklearn.pipeline.Pipeline	return clf	231
3	15	instance Pipeline	sklearn.pipeline.Pipeline	clf = Pipeline([('anova', anova_filter), ('svc...	231
4	16	instance Pipeline	sklearn.pipeline.Pipeline	clf.set_params(anova__k=20, svc__C=1e2)	231
5	17	instance Pipeline	sklearn.pipeline.Pipeline	return clf	231
6	25	class Pipeline	sklearn.pipeline.Pipeline	from sklearn.pipeline import Pipeline	319
7	37	instance Pipeline	sklearn.pipeline.Pipeline	classifier = Pipeline(steps=[('rbm', rbm), ('l...	319
8	42	instance Pipeline	sklearn.pipeline.Pipeline	classifier.fit(X_train, Y_train)	319
9	61	class Pipeline	sklearn.pipeline.Pipeline	from sklearn.pipeline import Pipeline	373
10	181	instance Pipeline	sklearn.pipeline.Pipeline	classifier = Pipeline(steps=[('rbm', rbm), ('l...	373
11	183	instance Pipeline	sklearn.pipeline.Pipeline	classifier.fit(X_train, Y_train)	373
12	358	class Pipeline	sklearn.pipeline.Pipeline	from sklearn.pipeline import Pipeline	682
13	364	instance Pipeline	sklearn.pipeline.Pipeline	clf = Pipeline(steps=[('rbm', rbm), ('logistic...	682
14	378	class Pipeline	sklearn.pipeline.Pipeline	from sklearn.pipeline import Pipeline	718

- We would be looking at the most used hyperparameter for each of the sklearn function.

```
Module :  AgglomerativeClustering
   Hyperparameter :  affinity  :  'euclidean'  :  7
   Hyperparameter :  connectivity  :  connectivity  :  1
   Hyperparameter :  linkage  :  'ward'  :  15
   Hyperparameter :  n_clusters  :  n_clusters  :  16
```

```
Module :  PCA
   Hyperparameter :  **kwargs  :  None  :  1
   Hyperparameter :  inputCol  :  "std_features"  :  2
   Hyperparameter :  copy  :  True  :  5
   Hyperparameter :  n_components  :  2  :  217
   Hyperparameter :  k  :  2  :  1
   Hyperparameter :  svd_solver  :  'full'  :  8
   Hyperparameter :  outputCol  :  "pca"  :  2
   Hyperparameter :  iterated_power  :  'auto'  :  1
   Hyperparameter :  random_state  :  42  :  9
   Hyperparameter :  tol  :  0.0  :  1
   Hyperparameter :  whiten  :  True  :  40
```

(More results and details can be found in the ipynb)

- Another file using pipeline (from imblearn) and test with instantiation analyser and jedi. “Odyssey_make_pipeline_imblearn.ipynb” file contains the analysis where jedi is used to extract pipeline used in the code apart from sklearn.