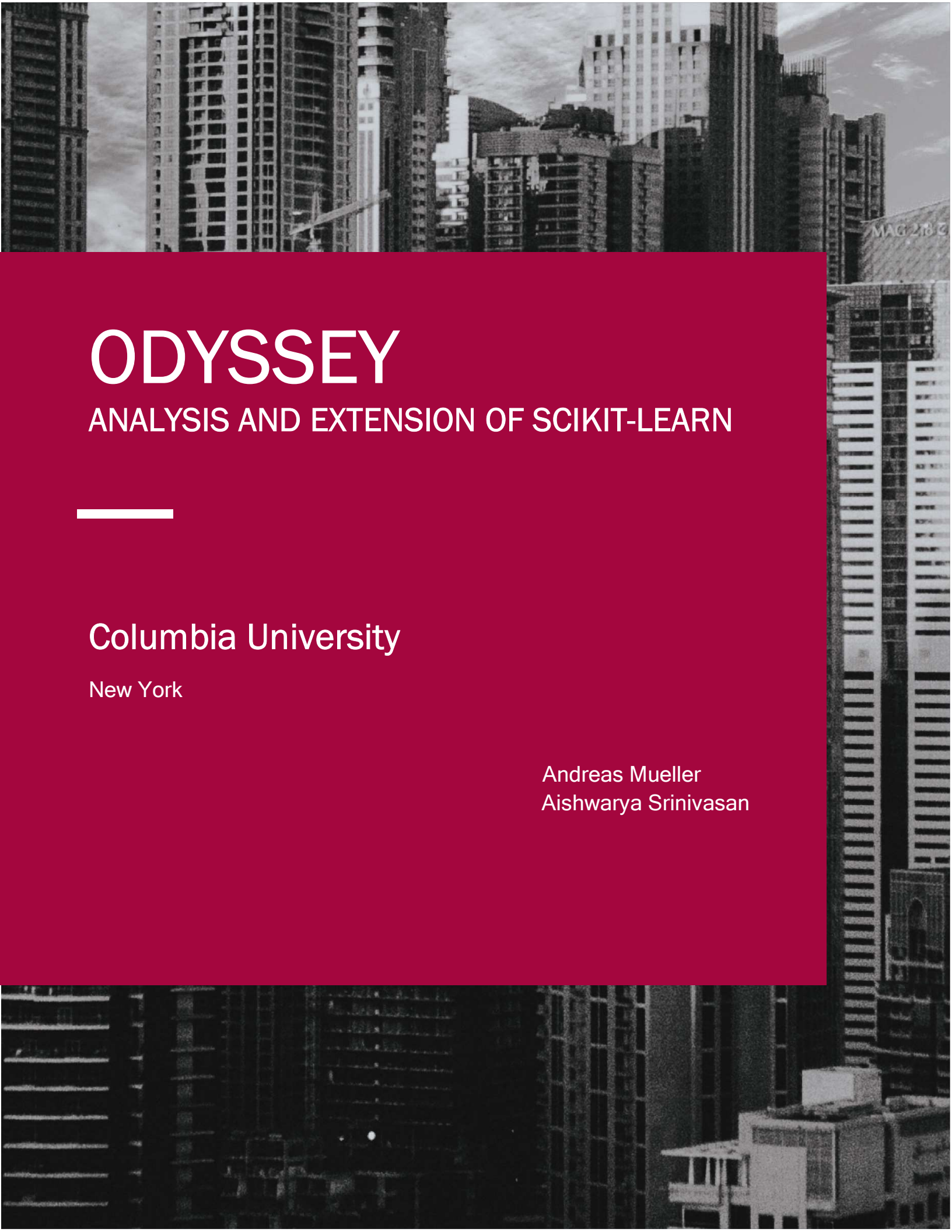

ODYSSEY

ANALYSIS AND EXTENSION OF SCIKIT-LEARN

Columbia University

New York

Andreas Mueller
Aishwarya Srinivasan

Introduction

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.

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 be building on the Odyssey library which was developed to support this project. (https://github.com/alan97/odyssey)

Methods

- For extracting the data from the open/public repositories in GitHub, Google BigQuery API is used with SQL queries.

- The python files extracted can be directly parsed, but the ipython notebooks are in json format, hence they need to be parsed and converted to python code format. For this, nbconvert package is used.

- The format of using the nbconvert is:
  ipython nbconvert notebook.ipynb --to script

- For the analysis, the scikit-learn classes, functions and sub-module usages are parsed using jedi parser and InstantiationAnalyzer parser.

As the initial roadmap for the analysis of the usage of scikit-learn, the following things needs to be explored:

- Comparing the usage of modules, functions and classes of scikit-learn.

- Comparing the usage across ipython and python files available in public repositories in Github extracted using Google BigQuery.

- Scikit-learn Pipeline functionality, which is similar to the make_pipeline functionality, both within pipeline module; which is also similar to make_pipeline inside imblearn. An important task is to make a program that builds a parse tree from the python code and understands each function call differently.

- For building a well-defined parse tree that recognizes data variables, functions being used, their linkage to modules and libraries. This gives a better insight to the library usages, if two functions with same name are used.

- Understanding the uses of hyperparameters for each function usages and analyzing the usage distribution.

Scikit-Learn Analysis – Description

- Odyssey Exploratory analysis

  - This notebook is built to extract data from the GitHub (repositories, python files and ipython notebooks) using the Google BigQuery API.
  - Upon extracting data, python files and notebooks containing scikit-learn imports are filtered.
  - Exploratory analysis on the number of files importing scikit-learn in each repository is plotted.
    Within each python file and notebook, the modules and functions of scikit-learn used are analyzed and plotted.

- Odyssey make_pipeline using jedi and Instantiation Analyser

  - The notebook contains a small python script which contains the use of make_pipeline.
  - Two parsers are applied to parse this code: InstantiationAnalyzer from the odyssey package and jedi (open source).
  - It is seen that the jedi parser tracks down the function import and links each function separately with the library it has been called from.
  - While InstantiationAnalyzer lacks that property. Hence, the InstantiationAnalyzer should be improved by using jedi parser.

- Odyssey_make_pipeline imblearn using jedi

  - The notebook contains a small python script which uses make_pipeline function from imblearn.
  - With jedi, the usage of the function is analysed, along with other variables, functions and modules; and stored as a data frame- including Description of the usage, Package/Module and Usage.

- Odyssey module usages with sklearn

  - The notebook is used to list out a set of modules which frequently used with scikit-learn.
  - Repositories containing python files and ipython notebooks are extracted which includes the use of these modules with scikit-learn.
  - Exploratory analysis of the usage of these modules with scikit-learn are stored and plotted.

- Odyssey sub-module analysis

    - The notebook contains the extraction of the submodules in scikit-learn, for the analysis of the submodules in python files and ipython notebooks.
    - It contains the analysis of the submodules used in python files versus ipython notebooks.

- Odyssey hyperparameter jedi – InstantiationAnalyzer

    - The notebook contains code to extract repos with files containing scikit-learn usages.
    - A list of all scikit-learn functions is extracted, as we require to analyze the hyperparameter usages of these functions.
    - All the extracted python files and ipython notebooks are saved locally to avoid multiple calls to Google BigQuery.
    - The hyperparameter usage for each function of scikit-learn inside the python files and ipython notebooks are extracted using the InstantiationAnalyzer (which is a part of Odyssey package).
    - The hyperparameters extracted are saved as a dictionary an dumped
    - into a pickle file for reusability and easy access in finding the hyperparameters for functions as keys of the dictionary.
    - Using a jedi parser, files containing pipeline is extracted and hyperparameter analysis is applied to get a new dictionary of hyperparameter_py_unique_after_jedi_Pipeline.

- Odyssey hyperparameter usages in sklearn functions

    - The notebook contains the extraction and cleaning of python files and ipython notebooks.
    - It also contains the visualization of the tree structure of the hyperparameter usage.

Conclusion

- The InstantiationAnalyzer is seen to be not appropriate for parsing python codes and creating a parse tree for analyzing the function and module usages. Whereas, jedi parser is seen to be very efficient and can track back to the library being used behind each function call.

- Jedi parser is also able to trace back to the values of the variables being using as variables in hyperparameters of various scikit-learn functions.

- The visualizations of the functions of scikit-learn usage in python and ipython notebooks give an overall distribution over the scikit-learn usage.

- The most frequently used hyperparameter values for scikit-learn functions are informative to know the user specifications. Most of which is using default hyperparameter values. A lot of python files contain GridSearchCV for hyperparameter tuning.

- It is known that scikit-learn module pipeline contains two functions namely make_pipeline and Pipeline, which perform similar operations. make_pipeline is also available in an extension of scikit-learn, imblearn. The jedi parser turned out to be very efficient in identifying the current library associated with the pipeline function usages.

- The project also included the analysis of the steps used in pipeline, for different pipeline functions. The results of these analysis can be found in corresponding notebooks mentioned above.