SemFORMS: Automatic Generation of Semantic Transforms By Mining Data Science Code

Ibrahim Abdelaziz , Julian Dolby , Udayan Khurana , Horst Samulowitz , Kavitha Srinivas
T.J. Watson Research Center, IBM Research
{ibrahim.abdelaziz1,kavitha.srinivas}@ibm.com, {samulowitz, dolby, ukhurana}@us.ibm.com

Abstract

Careful choice of feature transformations in a dataset can help predictive model performance, data understanding and data exploration. However, finding useful features is a challenge, and while recent Automated Machine Learning (AutoML) systems provide some limited automation for feature engineering or data exploration, it is still mostly done by humans. We demonstrate a system called SemFORMS (Semantic Transforms), which mines useful expressions for a dataset from access to a repository of code that may target the same dataset/similar dataset. In many enterprises, numerous data scientists often work on the same or similar datasets, but are largely unaware of each other's work. SemFORMS finds appropriate code from such a repository, and normalizes the code to be an actionable transform that can be prepended into any AutoML pipeline. We demonstrate SemFORMS operating over example datasets from the OpenML benchmarks where it sometimes leads to significant improvements in AutoML performance.

1 Introduction

Enterprises that perform data analytics and modeling have multiple data scientists who often work on the same type of datasets while being largely unaware of each other's work. Their code often contains a wealth of domain specific knowledge to wrangle, explore, and visualize the data. For example, Listing 1 shows a real code snippet trimmed for illustrative purposes. It reads tabular data in a function on Line 3 while Line 4 declares a new function where two new expressions are added to the input dataframe *houses_df*. The defined expressions in Lines 5 and 6 present manipulations on a dataset that are about factors that influence house prices. For example, the expression `total_bedrooms/total_rooms` calculates the square footage dedicated to bedrooms as a ratio of the total square footage; a useful factor in predicting house value.

Such code is a great repository of applied domain knowledge, which is locked up in individual scripts and unfortunately lacks means for sharing and reuse. SemFORMS (Semantic Transforms) is a system to leverage this knowledge; it mines code and automatically creates data transforms that a data scientist can drop into an AutoML pipeline for a new similar dataset.

Listing 1 Example data science code, with transforms

```
 1  import pandas as pd
 2  def read_df():
 3    return pd.read_csv('houses.csv')
 4
 5  def manipulate_df(houses_df):
 6      houses_df['beds_to_total'] =
    ↪   houses_df['total_bedrooms'] /
    ↪   houses_df['total_rooms']
 7      houses_df['popdf'] = houses_df['population' ] /
    ↪   houses_df['households']
 8
 9  def main():
10      h_df = read_df()
11      manipulate(h_df)
```

The key technical challenge in mining such transforms is that this requires sophisticated program analysis techniques to identify key objects (e.g., pandas dataframes), and several reads/writes from dataframes; that is, how data flows through the program. This is especially challenging for dynamically typed languages such as Python. SemFORMS has the machinery to perform such extraction; to our knowledge, no other analysis framework can extract such expressions in dynamically typed languages such as Python. Furthermore, we use the analysis framework to normalize expressions and generate lambda expressions that can be used within the popular `scikit-learn` pipeline.

Our results on OpenML datasets show that when scripts exist in open repositories such as GitHub, SemForms suggested expressions improved AutoML performance on 10/41 datasets. The value of a system such as SemFORMS is that because evaluation of transforms is entirely automated, transforms that do not improve overall performance can be discarded without any effort on part of the data scientist. SemFORMS system is available as an open source system[1].

2 System Overview

Figure 1 describes the overall SemFORMS system. Given a new dataset, the system queries code repositories such as GitHub with the table's metadata, specifically the table and

[1]https://github.com/wala/graph4code/tree/master/semForms

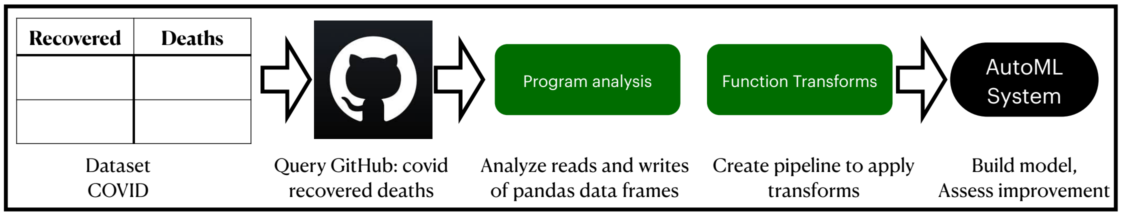
Recovered
Deaths
Program analysis
Function Transforms
AutoML System
Dataset COVID
Query GitHub: covid recovered deaths
Analyze reads and writes of pandas data frames
Create pipeline to apply transforms
Build model, Assess improvement

Figure 1: Overview of SemForms

column names. If the table is too wide, the system tries to search for relevant scripts using subsets of columns with the table name. These scripts are then fed into an analysis framework. Currently the analysis framework only operates on Python code; given that a large amount of data science code is written in Python, we targeted this language first, although nothing in the analysis framework is specific to Python.

The program analysis framework analyzes the Python code and identifies reads and writes to pandas dataframes, which are the normal heap structure that is modified in data preparation stages. Each possible write is a creation of a new column in the dataframe, and hence is of interest as a transform. The writes are analyzed and new code to reproduce that transform by eliminating differences in naming variables is generated by traversing the intermediate structures created by the analysis framework. New normalized code is generated as a lambda function so it can be used with the `scikit-learn's FunctionTransformer` class. For the purposes of the demo, we use these `FunctionTransformers` to create a pipeline with an estimator, and evaluate the advantage of adding the extracted transforms to the estimator compared to using only the original features.

3 Demonstration

We show in Listing 2 an example of how SemForms API is used for an example OpenML dataset (houses). We show the performance of traditional machine learning model on an OpenML dataset with and without SemForms recommended transforms. We propose to demonstrate SemForms functionality on any input dataset; an OpenML dataset or a local dataset that the user uploads. Once a dataset and its corresponding estimator (regressor or classifier) are chosen, our demonstration will show 1) the performance of the chosen estimator on the original dataset, 2) search and analysis of Github to extract a set of recommended transforms, and 3) automated application of these transforms on the performance of the estimator on the augmented data. We also show in Listing 3 the set of transforms for a completely different dataset – BNG(heart-h). At the conference, we will allow the audience to select an available dataset of their choice, and explore the dataset before and after augmentation.

4 Experiments

To assess the value of the transforms in terms of predictive performance, we used the state-of-the-art gradient boosted tree implementation LGBM [Ke *et al.*, 2017] as a strong baseline ML model. In order for LGBM to consume the input data, we performed a minimal set of standard prepossessing operations. Note that in the demo we use the RandomForest estimator just to demonstrate the API.

First, we measured the baseline performance of LGBM on 155 data sets. Among those, we are able to successfully apply transforms on 41 out of 155 benchmarks; that is only 41 benchmarks had any applicable scripts. We note that applicability of transforms to data is not just based on schema similarity, but data values as well; some transforms were not applicable because the data values were not present.

As shown in Table 1 shows we found relative performance improvements for 10/41 benchmarks in total. The underlying performance metric was `MSE` for regression targets, and `balanced accuracy` for classification problems.

Dataset	Relative Improvement	Dataset	Relative Improvement
BNG(heart-h)	12.8%	Houses	10.4%
BGG(heart-c)	5.8%	pc1	4.8%
Diabetes	3.3%	BNG(heart-c,nominal)	2.1%
pbcseq	1.9%	BNG(page-blocks)	0.8%
BNG(auto_price)	0.7%	strikes	0.6%

Table 1: Relative predictive performance gains on data sets. Regression targets used `MSE`, classification used `balanced accuracy`.

While not all improvements result in substantial gains in predictive performance, it is worth considering that: (a) improving a state-of-the-art ML model such as LGBM by solely augmenting the data automatically is not straightforward; as is highlighted by the significant computational effort needed to determine such features [Khurana *et al.*, 2016] and; (b) in most cases the applied transforms are semantically consistent and are therefore explainable. As shown in Table 2, when the system adds new features based on an expression such as `lambda df: numpy.logical_and((df['trestbps'] >120), (df['age'] >70))`, it actually enriches the data based on domain knowledge. It is possible that such relations may be determined automatically using transformation search-based approaches (e.g., [Khurana *et al.*, 2016]), but in all likelihood, they would require an exponentially expensive computational effort to find that one relation given that a combination of columns as well as values is required.

Adding new features to the data based on transformations can result in no improvement, or in the reduction of predictive performance, as we note in 31/41 benchmarks. Note, however, that because the entire process is automated, one can

Listing 2 SemForms API

```
# -------------------Input Dataset------------------#
dataset_name = 'houses' # can be OpenML or local
↪  dataset
dataset      = fetch_openml(dataset_name)
X            = dataset['data']
target       = dataset['target'].to_frame()

# -----------Performance on Original Data----------#
# Set standard classifier (could be any AutoML)
estimator = RandomForestRegressor(random_state = 1908)
scores = cross_val_score(estimator, X,
↪  numpy.ravel(target), cv=3, scoring='r2')
print("Averaged r2 score on (orig. data):  " +
↪  str(statistics.mean(scores)))
'''
Averaged r2 score on orig. data:  0.58
'''
# --------SemForms: Recommend new transforms-------#
# use GitHub API to find relevant Python notebooks
urls = search_github(dataset_name,
↪  dataset_cols=X.columns)
expressions = mine_code_for_expressions(urls)
# Analyze found transforms and if applicable, create
↪  SKLEARN Function Transforms as a pipeline
transforms_suggested, correlation =
↪  handle_transforms("both", expressions, target, X)
pretty_print_transforms(transforms_suggested)
'''
Filtered Expressions:
   lambda df: (df[ 'total_rooms' ] / df[ 'households'
↪  ])
   lambda df: (df[ 'total_bedrooms' ])
   lambda df: (df[ 'longitude' ] == 0)
   lambda df: (df[ 'longitude' ] > -114)
   lambda df: (df[ 'population' ] / df[ 'households'
↪  ])
'''
# ------SemForms: Apply recommended transforms-----#
# Add estimator to suggested transformation pipeline
transforms_suggested.append(('estimator', estimator))
pipeline = Pipeline(transforms_suggested)
# Evaluate with data augmentation added as function
↪  transformers based on original data
scores = cross_val_score(pipeline, X,
↪  numpy.ravel(target), cv=3, scoring='r2')
print("Averaged r2 score with SemForms:  " +
↪  str(statistics.mean(scores)))
'''
Averaged r2 score with SemForms:  0.65
'''
```

simply disregard the suggested transforms in searching for an optimal model, as is often the case when AutoML systems search optimal pipelines or hyperparameters.

5 Related Work

Code analysis: We need to track reads and writes to dataframes as they flow through the program, so we need analysis that is inter-procedural. Few systems such as Pysa[2] perform limited inter-procedural analysis. However, it has a single summary for a function, and none of the existing analysis systems, to our knowledge, provide the range of traditional machinery that we require here. To facilitate mining transforms, we extended GraphGen4Code [Abdelaziz *et al.*, 2021] to model heap access to track reads and writes to Python objects.

Mining from notebooks: Some works have targeted mining from notebooks to understand at a broad level, what methods or objects are most used using shallow analysis of variable names and imports such as [Rehman, 2019] or [Psallidas *et al.*, 2022], but none have the machinery needed to extract transforms as stated in [Rehman, 2019]. Amongst dynamic analysis based approaches, Auto-suggest [Yan and He, 2020] uses heuristics to handle hard issues such as missing data files and packages by searching. It depends on the availability of the CSV file in the public domain, and on dynamic analysis which is difficult to perform on a large number of scripts efficiently.

[2]https://pyre-check.org/docs/pysa-basics/

Listing 3 Suggested Expressions for BNG(heart-h)

```
'''
Filtered Expressions:
        lambda df: (df[ 'age' ] > 70)
        lambda df: (df[ 'sex' ])
        lambda df: (df[ 'thal' ])
        lambda df: (df[ 'sex' ] == 0)
        lambda df: (df[ 'sex' ] == 1)
        lambda df: (df[ 'oldpeak' ] < 4)
        lambda df: (df[ 'oldpeak' ] > 1)
        lambda df: (df[ 'age' ] > 75)
        lambda df: ((df[ 'slope' ]))
        lambda df: (df[ 'slope' ])
        lambda df: (df[ 'chol' ])
        lambda df: (df[ 'trestbps' ] > 120)
'''
```

Dataset	SemForm Suggested Expression
BNG(cleveland)	lambda df: (df['trestbps'] >120)
pc1	lambda df: (df['B'] >1)
BNG(cholesterol)	lambda df: numpy.logical_and((df['trestbps'] >120), (df['age'] >70))
houses	lambda df: (df['total_bedrooms'] / df['total_rooms'])
the-complete-pokemon	lambda df: (df['weight_kg'] / df['height_m'])
cleanedpokemon	lambda df: (df['type2'] == "flying")

Table 2: Examples of expressions with significant correlations for some OpenML and Kaggle datasets

Automated Feature Generation for Predictive Modeling: Existing automated feature engineering techniques typically explore the growth of given feature using pre-determined mathematical transformation functions which are agnostic of the nature of the data [Lam *et al.*, 2021; Kanter and Veeramachaneni, 2015; Kaul *et al.*, 2017; Le and Gulwani, 2014; Yan and He, 2020; He *et al.*, 2018; Jin *et al.*, 2017]. Our work is orthogonal to these systems where SemForms' recommended expressions can be automatically applied to input datasets as new features to improve ML models performance.

6 Conclusion

In this work, we introduced Semantic Transforms (SemFORMS), an approach for mining transforms from existing repositories to re-use prior work performed by other data scientists on similar datasets. Our demo highlights how the extracted features from code can be readily dropped into a `scikit-learn` pipeline and be re-used to improve performance.

References

[Abdelaziz *et al.*, 2021] Ibrahim Abdelaziz, Julian Dolby, James P McCusker, and Kavitha Srinivas. A toolkit for generating code knowledge graphs. *The Eleventh International Conference on Knowledge Capture (K-CAP)*, 2021.

[He *et al.*, 2018] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek Narasayya, and Surajit Chaudhuri. Transform-data-by-example (tde) an extensible search engine for data transformations. *Proceedings of the VLDB Endowment*, 11(10):1165–1177, 2018.

[Jin *et al.*, 2017] Zhongjun Jin, Michael R Anderson, Michael Cafarella, and HV Jagadish. Foofah: Transforming data by example. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 683–698, 2017.

[Kanter and Veeramachaneni, 2015] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.

[Kaul *et al.*, 2017] Ambika Kaul, Saket Maheshwary, and Vikram Pudi. Autolearn—automated feature generation and selection. In *2017 IEEE International Conference on data mining (ICDM)*, pages 217–226. IEEE, 2017.

[Ke *et al.*, 2017] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[Khurana *et al.*, 2016] Udayan Khurana, Deepak Turaga, Horst Samulowitz, and Srinivasan Parthasarathy. Cognito: Automated feature engineering for supervised learning. In *IEEE ICDM*, pages 1304–1307, 2016.

[Lam *et al.*, 2021] Hoang Thanh Lam, Beat Buesser, Hong Min, Tran Ngoc Minh, Martin Wistuba, Udayan Khurana, Gregory Bramble, Theodoros Salonidis, Dakuo Wang, and Horst Samulowitz. Automated data science for relational data. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2689–2692. IEEE, 2021.

[Le and Gulwani, 2014] Vu Le and Sumit Gulwani. Flashextract: A framework for data extraction by examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 542–553, 2014.

[Psallidas *et al.*, 2022] Fotis Psallidas, Yiwen Zhu, Bojan Karlas, Jordan Henkel, Matteo Interlandi, Subru Krishnan, Brian Kroth, Venkatesh Emani, Wentao Wu, Ce Zhang, Markus Weimer, Avrilia Floratou, Carlo Curino, and Konstantinos Karanasos. Data science through the looking glass: Analysis of millions of github notebooks and ml.net pipelines. *SIGMOD Rec.*, 51(2):30–37, jul 2022.

[Rehman, 2019] Mohammed Suhail Rehman. Towards understanding data analysis workflows using a large notebook corpus. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 1841–1843, New York, NY, USA, 2019. Association for Computing Machinery.

[Yan and He, 2020] Cong Yan and Yeye He. Auto-suggest: Learning-to-recommend data preparation steps using data science notebooks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554, 2020.