Bayesian AutoML for Databases via the InferenceQL Probabilistic Programming System

Ulrich Schaechtle[1] **Cameron Freer**[2] **Zane Shelby**[1] **Feras Saad**[2] **Vikash Mansinghka**[2]

[1]Digital Garage
[2]Massachusetts Institute of Technology

Abstract InferenceQL is a probabilistic programming system for scalable Bayesian AutoML from database tables. InferenceQL is designed to help make Bayesian approaches to data analysis more accessible to broad audiences and to assist experts in auditing and improving the quality of data, models, and inferences. Unlike traditional probabilistic programming systems, InferenceQL provides automation for learning models online using nonparametric Bayesian structure learning of probabilistic programs. Experts can override these models with custom probabilistic programs for specific subsets of variables and conditional distributions. For a broad class of models, InferenceQL can generate realistic synthetic data subject to constraints and can automatically compute exact probabilities and mutual information values. Finally, InferenceQL aims to enable scalable Bayesian model criticism via posterior predictive checks, data quality screening via conditional probability calculation, fairness auditing via conditional probability ratios, and synthetic data generation to enhance privacy. These capabilities are accomplished using constructs that interleave standard database queries with Bayesian inference.

Automated Bayesian inference from databases is important and useful in several ways. First, many real-world databases have high rates of missing values, more fields than observed records, heterogeneous data types, high rates of data entry error, and other factors that complicate the application of traditional ML-based AutoML techniques [9, App. E]. Second, many real-world applications benefit from uncertainty quantification, interactive model checking and model criticism, and conditional probability estimation for ad-hoc fairness auditing. These problems are naturally formulated in terms of Bayesian inference [11, 28].

InferenceQL is a probabilistic programming system for automated Bayesian inference from database tables. InferenceQL provides a domain-general mechanism for online Bayesian structure learning of probabilistic program source code, as well as domain-general mechanisms for scalable exact and approximate inference in these probabilistic programs. Users thus do not have to know how to write probabilistic programs in order to use InferenceQL to solve problems. Instead, users rely on automated data modeling techniques to navigate the design choices that might otherwise be handled by experienced modelers. InferenceQL also enables Bayesian inference operations to be interleaved with ordinary SQL operations, yielding complex database-native workflows for Bayesian AutoML. InferenceQL has been used successfully in field tests for a broad range of applications, including AutoML for clinical trial oversight in three real-world clinical trials.

This workshop paper introduces InferenceQL via an exploratory data analysis application. It also briefly reviews the system architecture of InferenceQL and the class of probabilistic programs that deliver its AutoML capabilities. It presents preliminary quantitative results from experiments comparing InferenceQL's modeling accuracy to GLM, VAE, and CTGAN baselines. Finally, it reviews related work, including both modeling formalisms and ML and database integrations, and discusses some limitations and broader impacts.

AutoML Conference 2022 Workshop Track

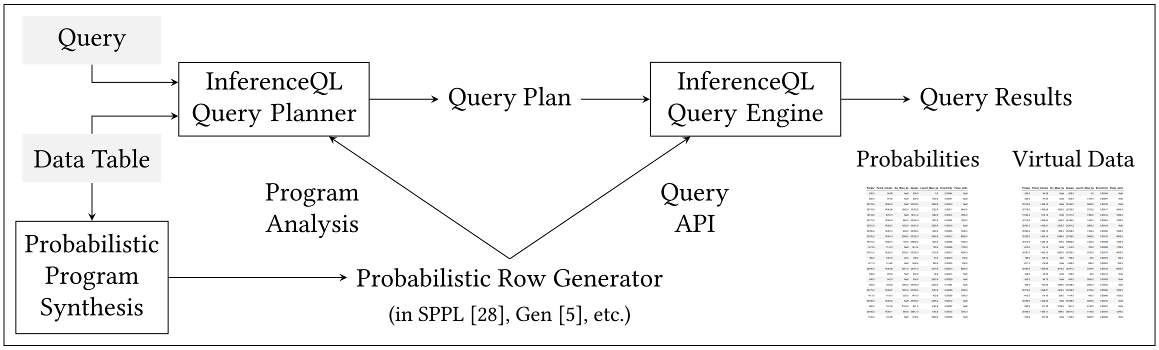


Figure 1: System architecture of InferenceQL.

Name	Country_of_Operator	Operator_Owner	Users	Purpose	Class_of_Orbit	Type_of_Orbit
Prometheus 1A	USA	Los Alamos Nati	Military	Technology Develo	LEO	Sun-Synchronous
Eutelsat 28A	Multinational	European Teleco	Commercial	Communications	GEO	NaN
SMDC-ONE 1.2	USA	U.S. Army Space	Military	Technology Develo	LEO	NaN
Lacrosse/Onyx	USA	National Reconn	Military	Surveillance	LEO	Intermediate
SMOS (Soil Mo	ESA	Centre National	Government	Earth Observation	LEO	Sun-Synchronous
Compass G-11	China (PR)	Chinese Defense	Military	Navigation/Global	GEO	NaN
Echostar 6	USA	Echostar Techno	Commercial	Communications	GEO	NaN
INMARSAT 4 F2	United Kingdom	INMARSAT, Ltd.	Commercial	Communications	GEO	NaN
Eutelsat 25C	Multinational	European Teleco	Commercial	Communications	GEO	NaN
Vinasat 2	Vietnam	Vietnamese Post	Government	Communications	GEO	NaN

Perigee_km	Apogee_km	Eccentricity	Period_minutes	Launch_Mass_kg	Dry_Mass_kg	Power_watts
500	506	0.00044	94.68	NaN	NaN	NaN
35788	35794	0.00007	1436.10	2950	1375	5900
483	789	0.02184	97.40	3	NaN	NaN
574	676	0.00729	97.21	14500	NaN	NaN
759	760	0.00007	100.00	658	630	1065
35776	35799	0.00027	1436.15	2300	NaN	NaN
35775	35798	0.00027	1436.12	3700	1493	11000
35773	35800	0.00032	1436.11	5458	NaN	13000
35780	35790	0.00012	1436.04	3170	1900	5900
35742	35776	0.00040	1434.69	2970	NaN	NaN

Date_of_Launch	Anticipated_Lifetime	Contractor	Launch_Site	Launch_Vehicle	longitude_radians	Inclination_radians
41597	NaN	Los Alamos Nation	Wallops Island Fl	Minotaur 1	NaN	0.707033
36958	12	Alcatel Space Ind	Guiana Space Cent	Ariane 5	0.498466	0.001222
41165	NaN	Miltec	Vandenberg AFB	Atlas 5	NaN	1.127483
36755	9	Lockheed Martin A	Vandenberg AFB	Titan IV	NaN	1.186824
40119	3	Thales Alenia Spa	Plesetsk Cosmodro	Breeze KM	NaN	1.717404
40963	8	Space Technology	Xichang Satellite	Long March 3A	1.029744	0.032638
36721	12	Lockheed Martin M	Cape Canaveral	Atlas 2 AS	-1.269029	0.001222
38664	15	EADS Astrium	Sea Launch (Odyss	Zenit 3SL	-0.920836	0.040666
37580	12	Alcatel Space Ind	Cape Canaveral	Atlas 2 AS	0.445059	0.000349
41044	15	Lockheed Martin C	Guiana Space Cent	Ariane 5 ECA	2.300344	0.001396

(a) Subset of satellites data table showing 21 variables and 10 records

```
# FIRST GROUP OF DEPENDENT VARIABLES
cluster_view_1 ~ categorical(
  {0: 0.945, 1: 0.02, 2: 0.01, ...})
if (cluster_view_1 == 0)
  Eccentricity ~ norm(0.002, 0.01)
elif (cluster_view_1 == 2)
  Eccentricity ~ norm(0.075, 0.015)
elif (cluster_view_1 == 3)
  Eccentricity ~ norm(0.028, 0.017)
...

# SECOND GROUP OF DEPENDENT VARIABLES
cluster_view_2 ~ categorical(
  {0: 0.45, 1: 0.365, 2: 0.01, ...})
if (cluster_view_2 == 0)
  Power_watts     ~ norm(870.32, 877.80)
  Launch_mass_kg  ~ norm(442.08, 528.63)
  Dry_mass_kg     ~ norm(362.45, 321.64)
  Period_miniutes ~ norm(101.67, 56.02)
  Perigee         ~ norm(683.49, 56.02)
  Apogee          ~ norm(742.68, 2411.91)
elif (cluster_view_2 == 1)
  Power_watts     ~ norm(7157.58, 4629.09)
  Launch_mass_kg  ~ norm(3870.96, 1417.09)
  Dry_mass_kg     ~ norm(1921.21, 762.07)
  Period_miniutes ~ norm(1435.63, 57.13)
  Perigee         ~ norm(35820.37, 1434.57)
  Apogee          ~ norm(35701.83, 2548.60)
...
```

(b) Synthesized row generator

Figure 2: Synthesizing probabilistic programs that model heterogeneously typed cross-sectional data.

1 Example

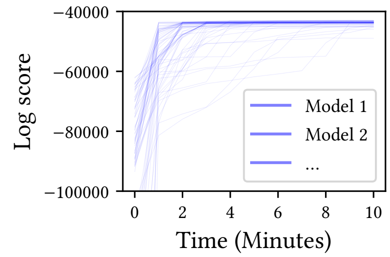
The InferenceQL system automates data analysis and machine learning tasks by allowing users to input data tables and queries and to automatically generate answers for them (Figure 1). It consists of a probabilistic program synthesis component [27] that creates generative model programs that are called *row generators*. The InferenceQL query planner and query engine use row generators to answer questions about the data and the domain by querying an underling probabilistic model.

Figure 2 shows an example of probabilistic program synthesis, which takes a heterogeneously-typed data table of satellites (maintained by the Union of Concerned Scientists [32]) and returns a probabilistic program that models the data. Figure 3(a) shows the high-level interface to creating synthesizing programs. Users can then compare synthetic data (generated from the probabilistic programs) with observed data in order to develop intuition about what the model learned from the data, shown in Figure 3(b). InferenceQL can generate virtual data from both marginal distributions and conditional distributions given a user-specified predicate (code box in Figure 3(b)). The two plots in Figure 3(b) illustrate a qualitative goodness-of-fit in the sense that distribution of synthetic (orange dots) samples appears to approximately match the observed data (black dots).

Data analysts can use the query language to search for probable anomalies and data-entry errors, shown in Figure 3(c). To find values for the column `Period_minutes` that the model considers

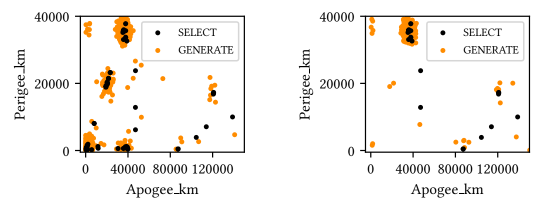
```
# Synthesize probabilistic
# programs given user-specified
# parameters in parameters.yaml
WITH LOAD 'parameters.yaml'
    AS params:
  BUILD default_model
  FOR data
  USING params;
```

```
#---- (parameters.yaml) ----#
# PARAMETERS FOR MODEL BUILDING
strategy: Gibbs # or SMC
number_models_in_ensemble: 100
inference_minutes: 10
override_schema:
  Perigee_km: numerical
  Purpose:    nominal
```



(a) Step 1: Synthesize probabilistic programs using the observed data table.

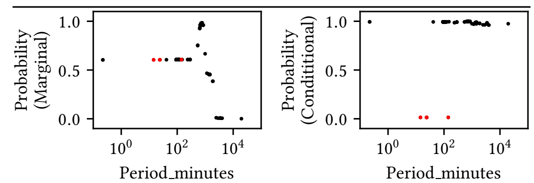
```
SELECT Perigee_km, Apogee_km FROM data INNER JOIN
  SELECT Perigee_km AS Perigee_km_generated, Apogee_km AS Apogee_km_generated
  FROM GENERATE Perigee_km, Apogee_km UNDER default_model;

SELECT Perigee_km, Apogee_km FROM data INNER JOIN
  SELECT Perigee_km AS Perigee_km_generated, Apogee_km AS Apogee_km_generated
  FROM GENERATE Perigee_km GIVEN Period_minutes < 1000
   UNDER default_model
WHERE Period_minutes < 1000;
```



(b) Step 2: Compare virtual data generated from the probabilistic programs to observed data.

```
WITH SELECT STDEV(Period_minutes) FROM data AS std_period:
  SELECT Period_minutes, Class_of_Orbit, Perigee_km, Apogee_km
  FROM data
  WHERE
      (PROBABILITY OF
        Period_minutes > (data.Period_minutes - std_period) AND
        Period_minutes < (data.Period_minutes + std_period)
          UNDER default_model)
      <
      (PROBABILITY OF
        Period_minutes > (data.Period_minutes - std_period) AND
        Period_minutes < (data.Period_minutes + std_period)
          GIVEN (* EXCEPT Period_minutes)
          UNDER default_model)
```

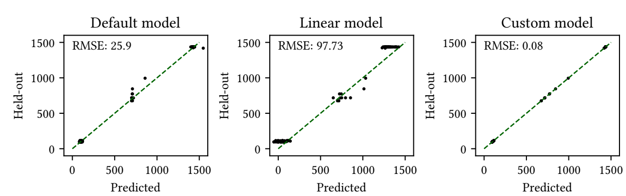
Period_minutes	Class_of_Orbit	Perigee_km	Apogee_km
23.9	GEO	35771.0	35805.0
142.0	GEO	35897.0	35909.0
14.3	GEO	35770.0	35803.0
23.9	GEO	35771.0	35805.0



(c) Step 3: Search for probable anomalies, which include data entry errors.

```
WITH SELECT * FROM data WHERE rowid <= 1000 AS training_data:
  BUILD default_model FOR training_data AND
  BUILD linear_model  FOR training_data AND

WITH SELECT * FROM data
  WHERE rowid > 1000 AS test_data:
    SELECT
      period,
      (PREDICT period GIVEN Perigee_km, Apogee_km UNDER
       default_model)
      (PREDICT period GIVEN Perigee_km, Apogee_km UNDER
       linear_model)
      (PREDICT period GIVEN Perigee_km, Apogee_km UNDER
       custom_model)
    FROM test_data;
```

```
function keplers_law(apogee, perigee)
  GM = 398600.4418; earth_radius = 6378;
  a = (abs(apogee) + abs(perigee)) * 0.5 + earth_radius;
  return 2 * π * sqrt(a^3 / GM) / 60 end;
@gen function custom_model(Perigee_km, Apogee_km)
  out = {:Period} ~ normal(keplers_law(Perigee_km, Aogee_km), .01)
  return out end;
```



(d) Step 4: Customize probabilistic programs using an orbital model from physics.

Figure 3: A representative data analysis workflow in InferenceQL on the satellites data.

improbable in light of the data, the query (left code box) produces the result by comparing the probability of the value for `Period_minutes` marginally and conditionally in the `WHERE` clause. The only rows returned are those whose conditional probability is lower than the marginal; the corresponding `Period_minutes` values are highlighted in red in the table and plots of Figure 3(c).

Finally, users with domain expertise can customize probabilistic programs. Figure 3(d) shows an example custom orbital model from physics. To quantitatively assess the goodness-of-fit, we first split the data into training and test data and build three models: the automatically synthesized default model, a generalized linear model (GLM), and a custom probabilistic program for noisy orbital physics. We then predict a column in the held-out data set. The default model predicts more accurately than the GLM (4x more accurate via root mean square error (RMSE)) and the custom probabilistic program beats the default (a further 300x improvement in RMSE).

Table 1: Virtual data benchmark.

	Jensen-Shannon Divergence to True Data			
Dataset	InferenceQL	CTGAN	Copulas	TVAE
Nursery	0.04	0.14	0.29	0.05
Tumor	**0.06**	0.40	0.20	0.45
Flare	**0.05**	0.22	0.23	0.28
Car	0.05	0.16	0.12	0.08
Mushroom	**0.08**	0.15	0.33	0.11
Soybean	**0.10**	0.18	0.22	0.36
Breast-cancer	**0.15**	0.38	0.43	0.38
Heart-disease	**0.08**	0.16	0.30	0.44
Connect-4	**0.04**	0.10	0.22	0.08
Chess	**0.03**	0.10	0.17	0.05

Table 2: Anomaly detection benchmark.

		Anomaly Detection Accuracy	
Dataset	Target	InferenceQL	GLM
Abalone	Rings	86%	82%
Breast-cancer	class	100%	60%
Heart-disease	num	97%	47%

Table 3: Runtime optimization benchmark.

		InferenceQL (SPPL backend)		**Python API (SPPL)**
Dataset	Target	Independence Analysis	Default Optimization	Default Optimization
Nursery	Evaluation	11.14 ± 7.31	501.35 ± 571.08	302.29 ± 366.1
Tumor	Type	1.99 ± 0.34	3.21 ± 0.51	3.24 ± 0.8
Flare	Num_common_flares	6.96 ± 2.56	14.32 ± 14.27	8.84 ± 7.78
Car	Evaluation	13.03 ± 4.69	153.91 ± 275.4	92.3 ± 158.98
Mushroom	Edible?	31.34 ± 6.17	34.07 ± 6.87	24.74 ± 5.78
Soybean	Disease	9.44 ± 3.05	11.7 ± 2.26	9.05 ± 2.19
Breast-cancer	Diagnosis	5.07 ± 0.71	6.78 ± 0.73	3.93 ± 0.82
Heart-disease	Present?	3.49 ± 1.49	11.61 ± 8.11	8.99 ± 6.02
Connect-4	White_can_win	34.24 ± 24.61	65.26 ± 59.99	44.61 ± 36.0
Chess	Outcome	61.83 ± 45.34	86.27 ± 51.3	62.79 ± 34.56

2 Experiments

We now report experiments evaluating InferenceQL against statistical and neural baselines.

Virtual data benchmark. Table 1 shows the average Jensen-Shannon divergence between virtual and observed data for all pairwise marginals in 10 datasets from the UCI machine learning repository [8], according to simulations from InferenceQL, Gaussian copulas [23], CTGAN [34], and TVAE [34]. The bold entries indicate statistically significant lowest error under a Bonferroni corrected Mann-Whitney U test, which are achieved by InferenceQL in 8 of 10 benchmark problems and zero times by other techniques.

Anomaly detection benchmark. Table 2 shows that InferenceQL detects a higher percentage of anomalies than does a GLM baseline on three datasets from the UCI repository. Anomalies were inserted into a target column by flipping the class label in each row with probability 0.05 and detected using a query similar to the one in Figure 3(d).

Query optimization. The InferenceQL query planner contains a built-in optimization for querying row generators specified in the SPPL language [28]. Table 3 shows the runtime of InferenceQL queries for computing the conditional probability of all cell values in one target column given all the other values in the same row, for 10 datasets from the UCI repository. The third column shows the runtime using InferenceQL's independence analysis optimization, which statically eliminates from the query all conditioning variables that are structurally independent of the target variable. The fourth column shows the runtime using InferenceQL without independence analysis and the final column shows the runtime using the Python API to SPPL, which both do not automatically leverage independence analysis and are slower in cases where independencies can be exploited.

3 Related Work

Various AutoML systems have been developed for tabular data, many of which focus on solving a specific discriminative problem (e.g., regression or classification); whereas in InferenceQL the user automatically obtains a generative model that can be queried repeatedly to answer a wide range of questions about the data. These include Amazon's AutoGluon-Tabular [9] and SageMaker Autopilot [7], Google Cloud Platform AutoML Tables [17], Uber's Ludwig [20], H2O AutoML [16], and a number of earlier systems such as Auto-WEKA [31], auto-sklearn [10], hyperopt-sklearn [2], TPOT [21], autoxgboost [30], ML-Plan [19], OBOE [35], GAMA [12], and Auto-Keras [15]. A survey and comparison of many of these systems can be found in Erickson et al. [9, §3].

InferenceQL is most similar to the BayesDB [18] probabilistic programming system. Like InferenceQL, BayesDB requires the underlying models to satisfy the composable generate population model interface [24]. InferenceQL aims to support new queries that include predicates as constraints and custom models in the Gen probabilistic programming language [5].

InferenceQL makes use of sum-product expressions [28], which are a type of of probabilistic circuit [6]. These symbolic representations are designed to to balance modeling expressiveness with tractable inference. Previous database systems have incorporated probabilistic circuit representations to speed up queries [13, 22, 33] outside of a probabilistic programming context.

Probabilistic databases [29] are a class of systems that take a different approach from InferenceQL, where they typically assign weights to facts in a database and implement queries that compute the probability of Boolean formulas. Unlike InferenceQL, standard probabilistic databases do not include automatic probabilistic model discovery from high-dimensional and heterogeneously-typed data tables. Another probabilistic approach to database systems and query languages involves augmenting relational algebras with techniques that enable predictive modeling, learning, and inference. Relational algebras have been extended by functions for imputation [4], time series prediction [1], random data generation [14] and simulation [3].

4 Conclusion

Limitations. A central limitation of InferenceQL is the lack of large-scale evaluations of its accuracy and performance that span a range of different data regimes. In principle, InferenceQL should be able to match the accuracy of many real-world ML deployments by using hybrids of generative models with decision-tree classifiers (as implemented in its SPPL backend), though it remains to be seen how effective and competitive with other AutoML approaches this approach is in practice. Also, once users leave the fully-automated setting (e.g., to customize or override the learned probabilistic programs), the InferenceQL API for custom models as in Figure 3(d) can be challenging to implement correctly. Finally, InferenceQL only applies to cross-sectional data tables; it would be worthwhile to extend InferenceQL with domain-general Bayesian structure learning methods for multivariate time series [25] or relational systems [26].

Broader Impact. One potential benefit of InferenceQL is educational. If it turns out (much like SQL) to be learnable by many end-users, then InferenceQL has the potential to increase access to and literacy in Bayesian reasoning for a broad audience. InferenceQL also has the potential to reduce the marginal cost of Bayesian data analysis for applied practitioners, which drives both beneficial impact and risks of harm. Benefits include improved analytics and decision-making capabilities in the public interest, especially in applications where realtime learning is valuable, such as adaptive design of scientific experiments. The risks include reduced cost for invasion of privacy, as well as analytics and decision-making towards other harmful aims.

References

[1] Anish Agarwal, Abdullah Alomar, and Devavrat Shah. tspDB: Time series predict DB. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 27–56. PMLR, 2021.

[2] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: A python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015. doi:10.1088/1749-4699/8/1/014008/meta.

[3] Zhuhua Cai, Zografoula Vagena, Luis Perez, Subramanian Arumugam, Peter J. Haas, and Christopher Jermaine. Simulation of database-valued Markov chains using SimSQL. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 637–648. ACM, 2013. doi:doi/10.1145/2463676.2465283.

[4] José Cambronero, John K. Feser, Micah J. Smith, and Samuel Madden. Query optimization for dynamic imputation. *Proceedings of the VLDB Endowment*, 10(11):1310–1321, 2017. doi:10.14778/3137628.3137641.

[5] Marco F. Cusumano-Towner, Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 221–236. ACM, 2019. doi:10.1145/3314221.3314642.

[6] Adnan Darwiche. Tractable boolean and arithmetic circuits. In Pascal Hitzler and Md Kamruzzaman Sarker, editors, *Neuro-Symbolic Artificial Intelligence: The State of the Art*, volume 342 of *Frontiers in Artificial Intelligence and Applications*, chapter 6, pages 146–172. IOS Press Ebooks, 2021. doi:10.3233/FAIA210353.

[7] Piali Das, Nikita Ivkin, Tanya Bansal, Laurence Rouesnel, Philip Gautier, Zohar Karnin, Leo Dirac, Lakshmi Ramakrishnan, Andre Perunicic, Iaroslav Shcherbatyi, Wilton Wu, Aida Zolic, Huibin Shen, Amr Ahmed, Fela Winkelmolen, Miroslav Miladinovic, Cedric Archembeau, Alex Tang, Bhaskar Dutt, Patricia Grao, and Kumar Venkateswar. Amazon SageMaker Autopilot: a white box AutoML solution at scale. In *Proceedings of the 4th International Workshop on Data Management for End-to-End Machine Learning*. ACM, 2020. doi:10.1145/3399579.3399870.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[9] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-Tabular: robust and accurate AutoML for structured data. *arXiv*, 2003.06505, 2020. doi:10.48550/arXiv.2003.06505.

[10] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[11] Andrew Gelman, Aki Vehtari, Daniel Simpson, Charles C. Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák. Bayesian workflow. *arXiv*, 2011.01808, 2020. doi:10.48550/arXiv.2011.01808.

[12] Pieter Gijsbers and Joaquin Vanschoren. GAMA: Genetic automated machine learning assistant. *Journal of Open Source Software*, 4(33):1132, 2019. doi:10.21105/joss.01132.

[13] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. DeepDB: learn from data, not from queries! *Proceedings of the VLDB Endowment*, 13(7):992–1005, 2020. doi:10.14778/3384345.3384349.

[14] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Perez, Chris Jermaine, and Peter J. Haas. The Monte Carlo database system: Stochastic analysis close to the data. *ACM Transactions on Database Systems*, 36(3), 2011. doi:10.1145/2000824.2000828.

[15] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-Keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019. doi:10.1145/3292500.3330648.

[16] Erin LeDell and Sebastien Poirier. H2O AutoML: Scalable automatic machine learning. In *Proceedings of the 7th ICML Workshop on AutoML*, 2020.

[17] Yifeng Lu. An end-to-end AutoML solution for tabular data at KaggleDays, 2019. URL `https://ai.googleblog.com/2019/05/an-end-to-end-automl-solution-for.html`.

[18] Vikash K. Mansinghka, Richard Tibbetts, Jay Baxter, Pat Shafto, and Baxter Eaves. BayesDB: a probabilistic programming system for querying the probable implications of data. *arXiv*, 1512.05006, 2015. doi:10.48550/arXiv.1512.05006.

[19] Felix Mohr, Marcel Wever, and Eyke Hüllermeier. ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107(8):1495–1515, 2018. doi:10.1007/s10994-018-5735-z.

[20] Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala. Ludwig: A type-based declarative deep learning toolbox. *arXiv*, 1909.07930, 2019. doi:10.48550/arXiv.1909.07930.

[21] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. Automating biomedical data science through tree-based pipeline optimization. In *Applications of Evolutionary Computation*, volume 9597 of *Lectures Notes in Computer Science*, pages 123–137. Springer, 2016. doi:10.1007/978-3-319-31204-0_9.

[22] Dan Olteanu and Maximilian Schleich. Factorized databases. *SIGMOD Record*, 45(2):5–16, 2016. doi:10.1145/3003665.3003667.

[23] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *Proceedings of the 3rd IEEE International Conference on Data Science and Advanced Analytics*, pages 399–410. IEEE, 2016. doi:10.1109/DSAA.2016.49.

[24] Feras Saad and Vikash K. Mansinghka. A probabilistic programming approach to probabilistic data analysis. In *Advances in Neural Information Processing Systems*, pages 2011–2019. Curran Associates, Inc., 2016.

[25] Feras A. Saad and Vikash K. Mansinghka. Temporally-reweighted Chinese restaurant process mixtures for clustering, imputing, and forecasting multivariate time series. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 755–764. PMLR, 2018.

[26] Feras A. Saad and Vikash K. Mansinghka. Hierarchical infinite relational model. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1067–1077. PMLR, 2021.

[27] Feras A. Saad, Marco F. Cusumano-Towner, Ulrich Schaechtle, Martin C. Rinard, and Vikash K. Mansinghka. Bayesian synthesis of probabilistic programs for automatic data modeling. *Proceedings of the ACM on Programming Languages*, 3(POPL), 2019. doi:10.1145/3290350.

[28] Feras A. Saad, Martin C. Rinard, and Vikash K. Mansinghka. SPPL: Probabilistic programming with fast exact symbolic inference. In *PDLI 2021: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Design and Implementation*, pages 804–819. ACM, 2021. doi:10.1145/3453483.3454078.

[29] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Number 16 in Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. doi:10.2200/S00362ED1V01Y201105DTM016.

[30] Janek Thomas, Stefan Coors, and Bernd Bischl. Automatic gradient boosting. In *Proceedings of the International Workshop on Automatic Machine Learning*, 2018.

[31] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 847–855. ACM, 2013. doi:10.1145/2487575.2487629.

[32] Union of Concerned Scientists. UCS satellite database, 2016. URL `https://www.ucsusa.org/resources/satellite-database`.

[33] Guy Van den Broeck and Dan Suciu. Query processing on probabilistic data: A survey. *Foundations and Trends® in Databases*, 7(3-4):197–341, 2017. doi:10.1561/1900000052.

[34] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[35] Chengrun Yang, Yuji Akimoto, Dae Won Kim, and Madeleine Udell. OBOE: Collaborative filtering for AutoML model selection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1173–1183. ACM, 2019. doi:10.1145/3292500.3330909.