SNOWFLAKE CORTEX

MASTERCLASS
2024
HANDS-ON!

ROADMAP TO SNOWFLAKE CORTEX (2023)

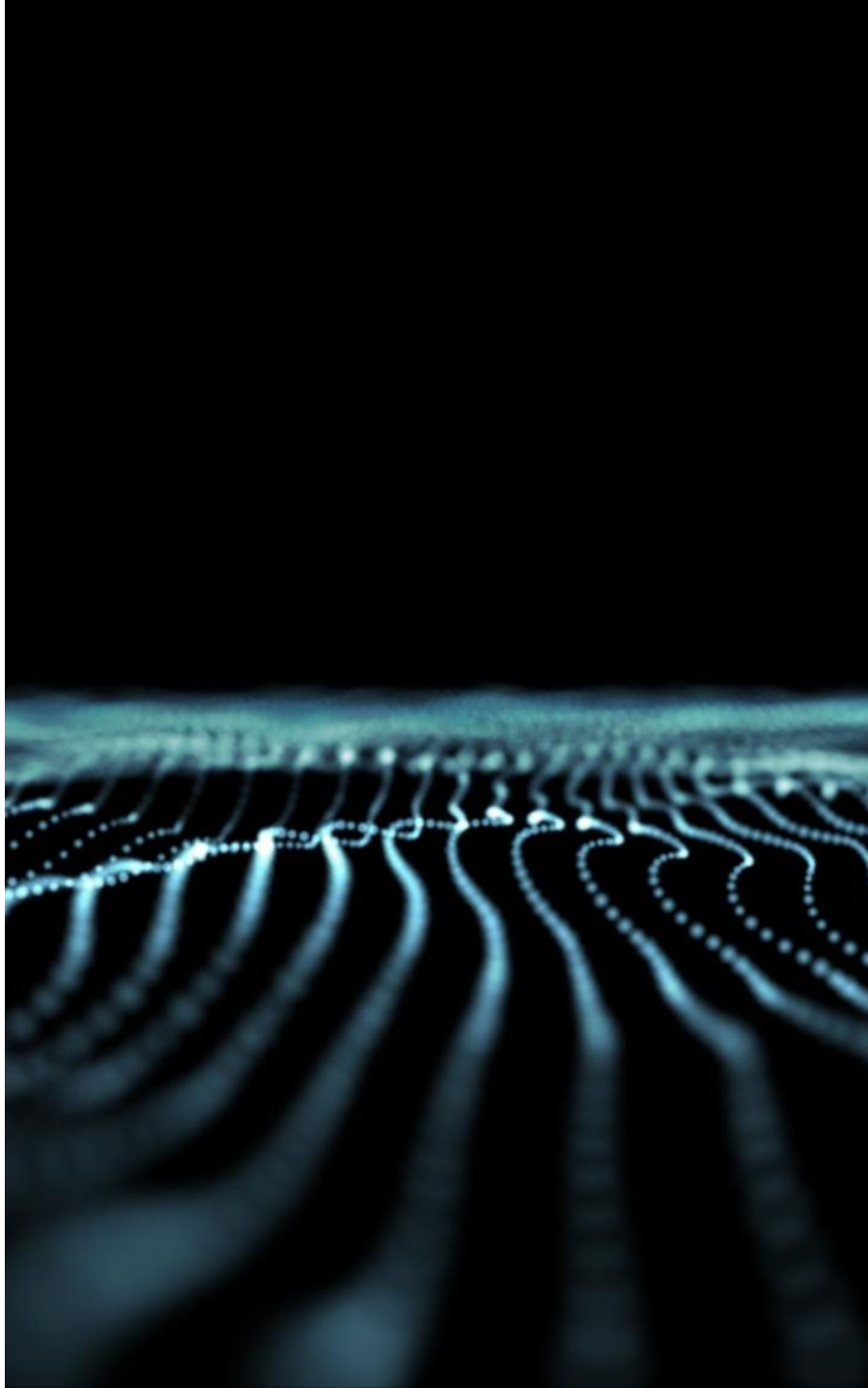
Jan 2023	Feb 2023	Mar 2023	Apr 2023
Object Dependencies	Access History Snowflake Alerts	Snowpipe Streaming Python Worksheets	Account Replication
May 2023	**Jun 2023**	**Jul 2023**	**Aug 2023**
Snowpark ML Modeling ML-Powered Functions Native App Framework	Snowpark ML Contribution Explorer Schema Inference	Container Services Dynamic Tables	Vectorized UDTFs
Sep 2023	**Oct 2023**	**Nov 2023**	**Dec 2023**
External Network Access	Budgets	Snowflake Notebooks Feature Store Iceberg Tables	Time-Series Forecasting Anomaly Detection GPUs for SPCS

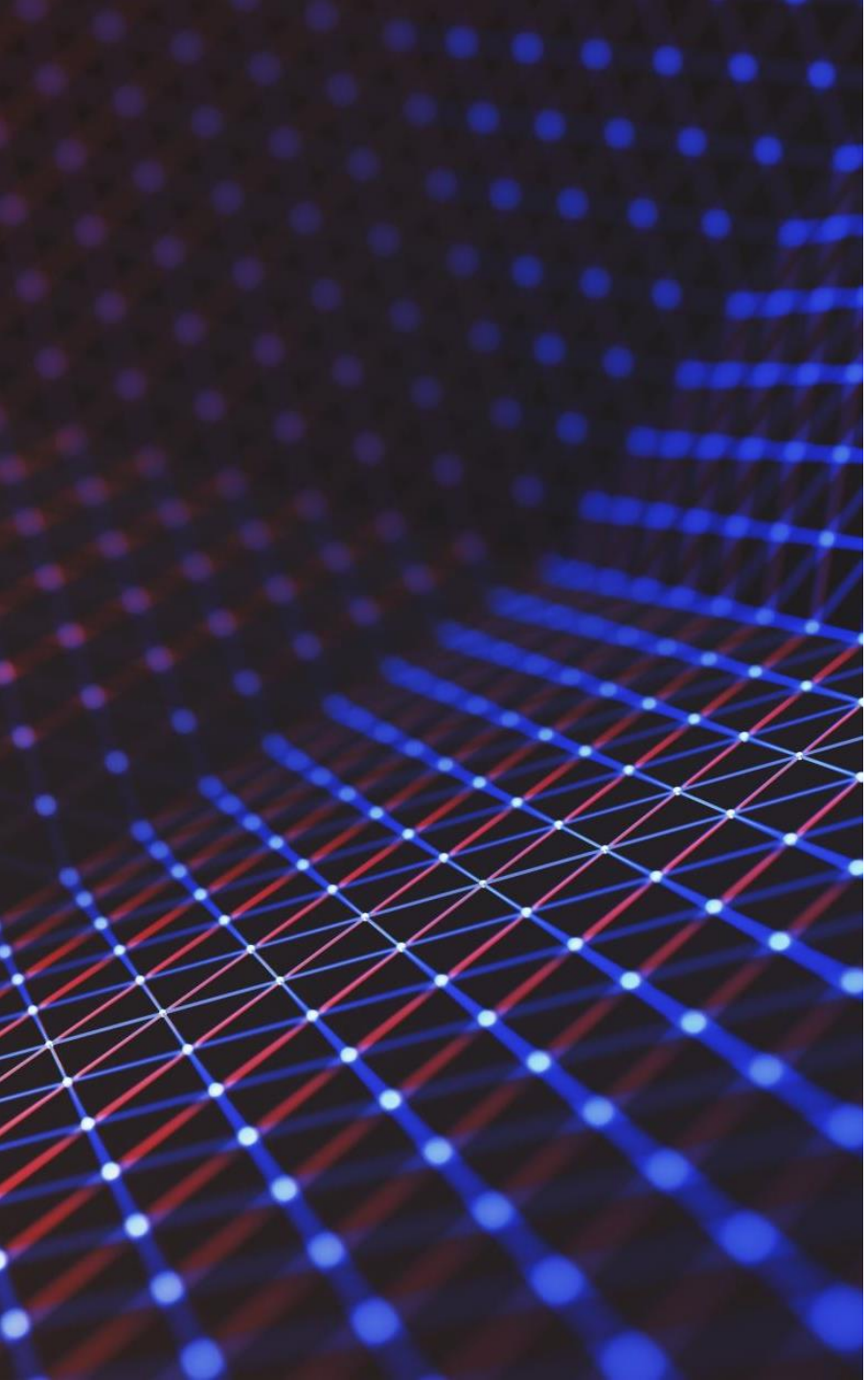
ROADMAP TO SNOWFLAKE CORTEX (2024)

Jan 2024	Feb 2024	Mar 2024	Apr 2024
Model Registry	Universal Search Hybrid Tables	LLM Functions Classification	Snowflake Copilot Snowflake Arctic Snowflake CLI
May 2024	**Jun 2024**	**Jul 2024**	**Aug 2024**
Sep 2024	**Oct 2024**	**Nov 2024**	**Dec 2024**

RELATED AI/ML SNOWFLAKE FEATURES

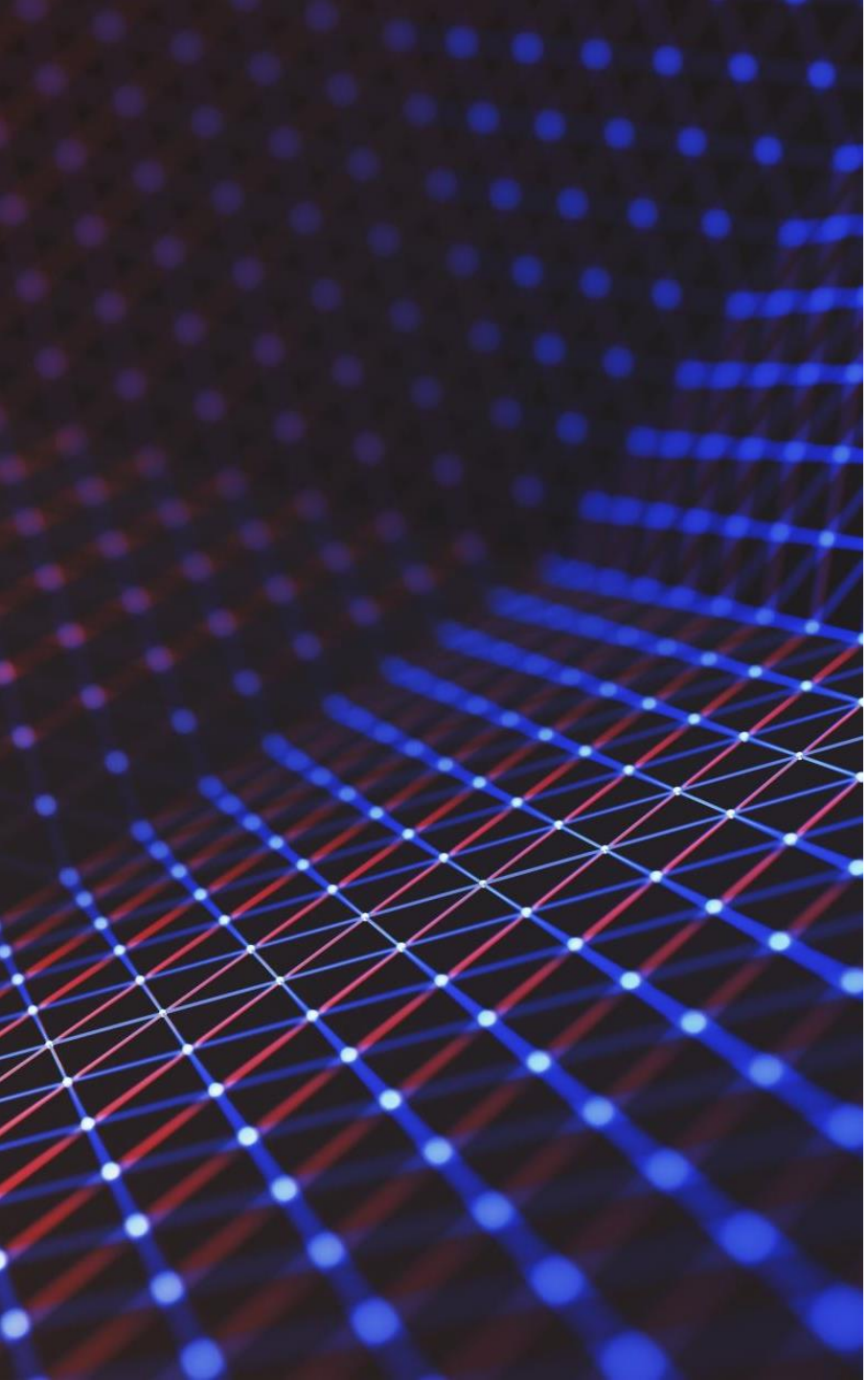
- **Snowpark DataFrame API** → since 2021+
- **Snowpark Functions & Procedures** → @sproc/udf/udtf…
- **External Access Integration** → to connect with external REST APIs and LLMs like OpenAI/ChatGPT…
- ***Snowflake Container Services (SPCS)** → in PrPr, to train LLMs / DL models w/ GPUs
- ***Snowflake Notebooks** → in PrPr, to run SQL/Python in Snowsight
- ***Snowflake Feature Store** → in PrPr, will likely be part of Snowflake MLOps





SNOWFLAKE CORTEX INFLUENCES

- Jupyter Notebooks
- Scikit-Learn (Sklearn)
- TensorFlow(+Keras)/PyTorch/MXNet
- SageMaker/Azure ML/Vertex AI
- PySpark DataFrame ← Apache Spark
- MLlib ← Apache Spark, Databricks
- BigQuery ML ← Google
- ChatGPT ← OpenAI API



SNOWFLAKE ACQUISITIONS

- **Streamlit** - 2022, $800M → Python apps
- **Applica** - 2022 → LLM for Document AI
- **Neeva** - 2023, $185M → Universal Search
- **Mistral** - 2023, investment → hosted LLMs
- **Ponder** - 2023 → Python for enterprise
- **Samooha** - 2023 → Data Clean Rooms

SNOWFLAKE CORTEX

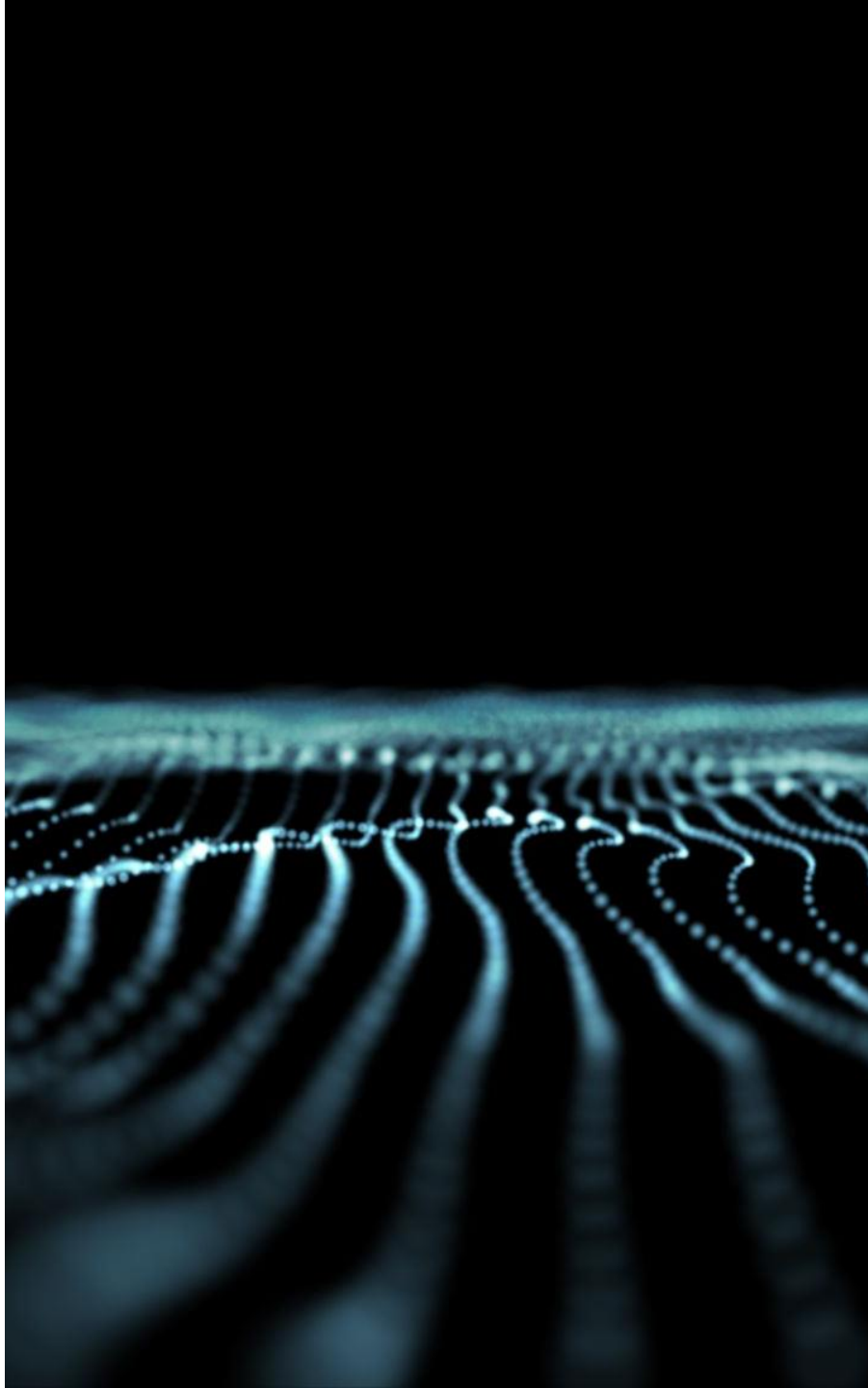
Snowpark ML

ML-Powered Functions

LLM Functions

SNOWFLAKE CORTEX

- **Snowpark ML APIs**
  - Snowpark ML Modeling
  - Snowpark ML Operations (MLOps) - w/ Model Registry
  - Snowpark ML Data Access (+Framework Connectors)
- **ML-Powered Functions (Classes)**
  - Time-Series Forecasting
  - Anomaly Detection
  - Contribution Explorer
  - Classification
- **LLM Functions (and UI Extensions)**
  - COMPLETE + EXTRACT_ANSWER
  - SENTIMENT + SUMMARIZE + TRANSLATE
  - **Snowflake Copilot + Universal Search + Document AI**



(1) SNOWPARK ML APIS

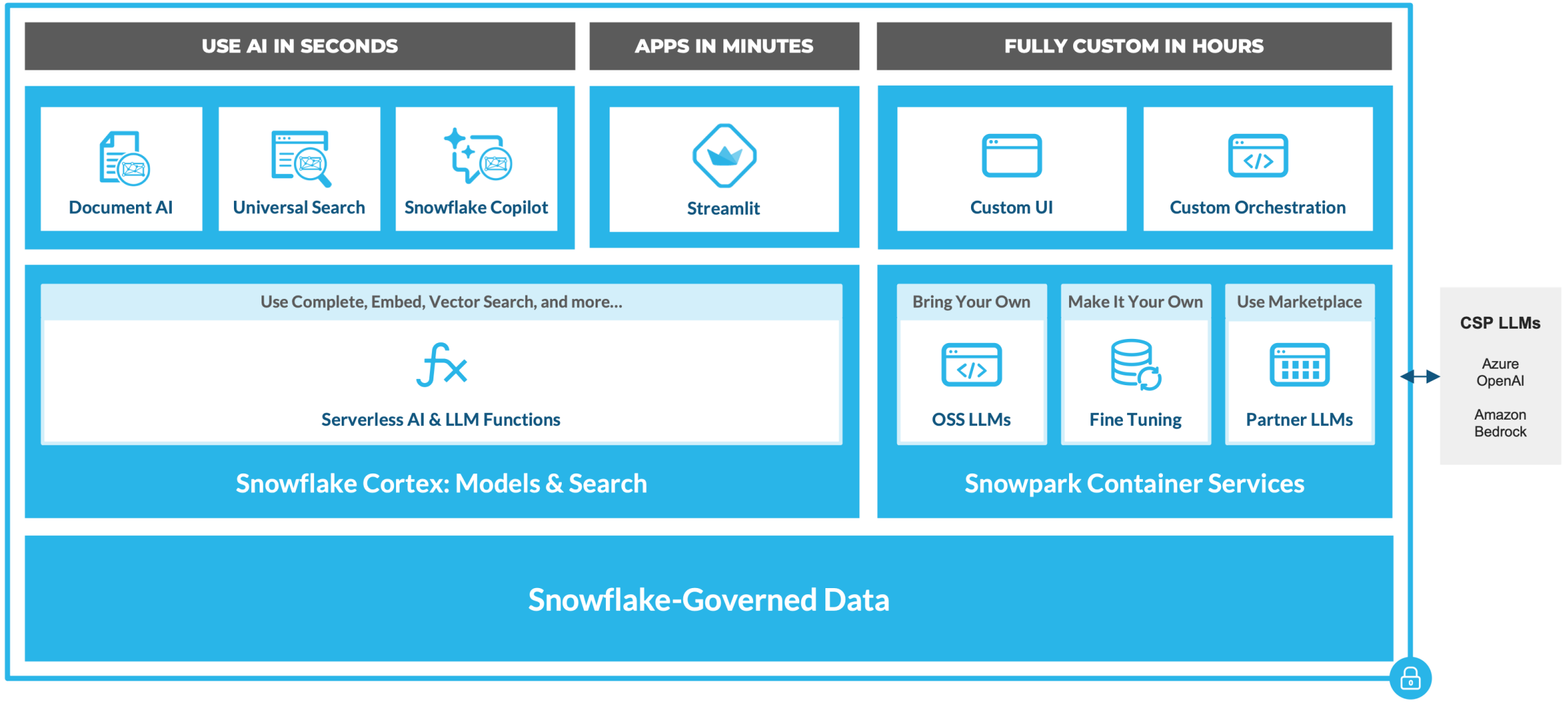
Modeling

Operations (MLOps)

Data Access

(2) ML-POWERED FUNCTIONS AND CLASSES

- Time-Series Forecasting
- Anomaly Detection
- Classification
- Contribution Explorer


USE AI IN SECONDS
APPS IN MINUTES
FULLY CUSTOM IN HOURS
Document AI
Universal Search
Snowflake Copilot
Streamlit
Custom UI
Custom Orchestration
Use Complete, Embed, Vector Search, and more...
Serverless AI & LLM Functions
Snowflake Cortex: Models & Search
Bring Your Own
Make It Your Own
Use Marketplace
OSS LLMs
Fine Tuning
Partner LLMs
Snowpark Container Services
CSP LLMs
Azure OpenAI
Amazon Bedrock
Snowflake-Governed Data

(3) LLM FUNCTIONS

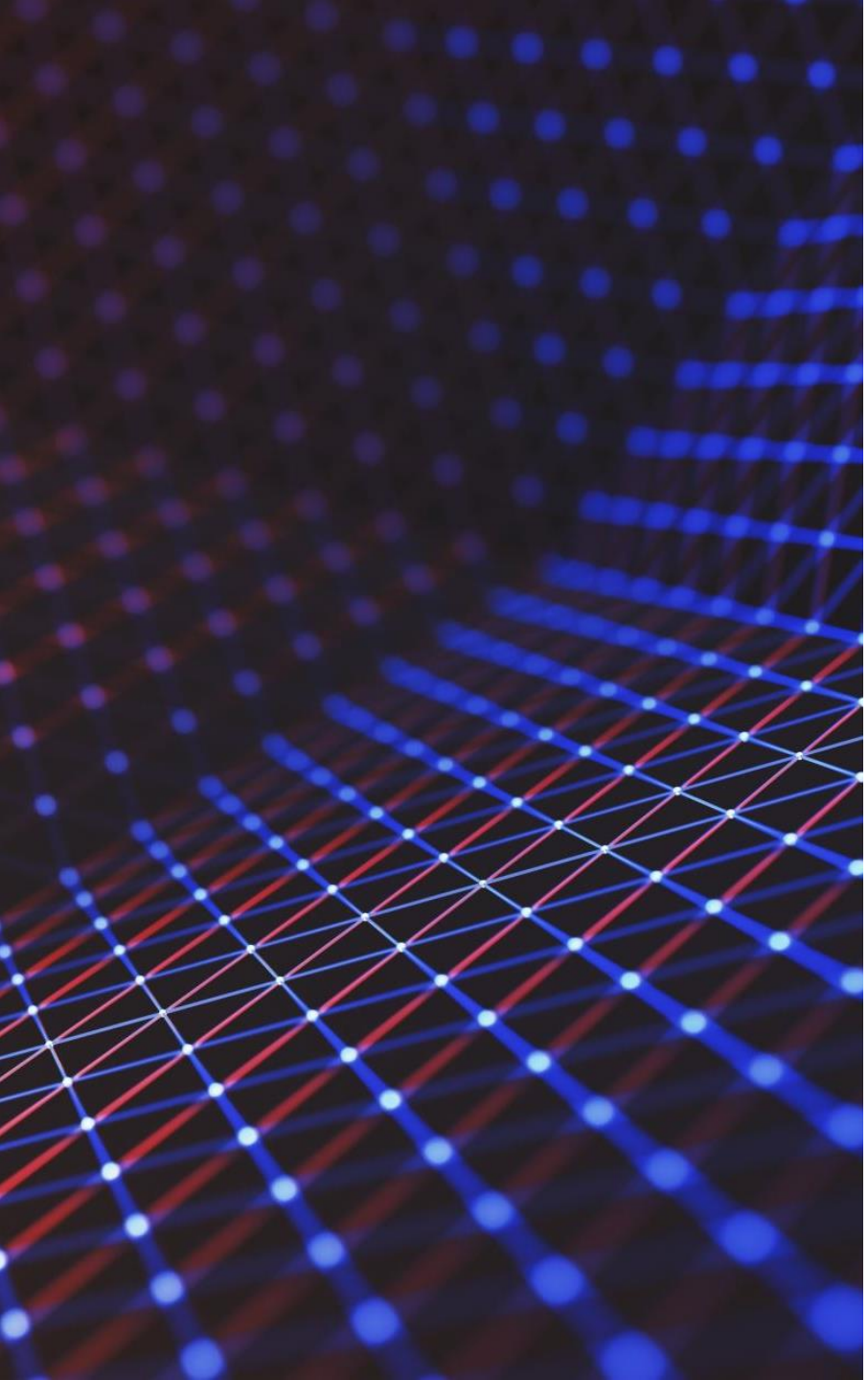
- COMPLETE
- EXTRACT_ANSWER
- SENTIMENT
- SUMMARIZE
- TRANSLATE

(4) LLM UI EXTENSIONS

Snowflake Copilot

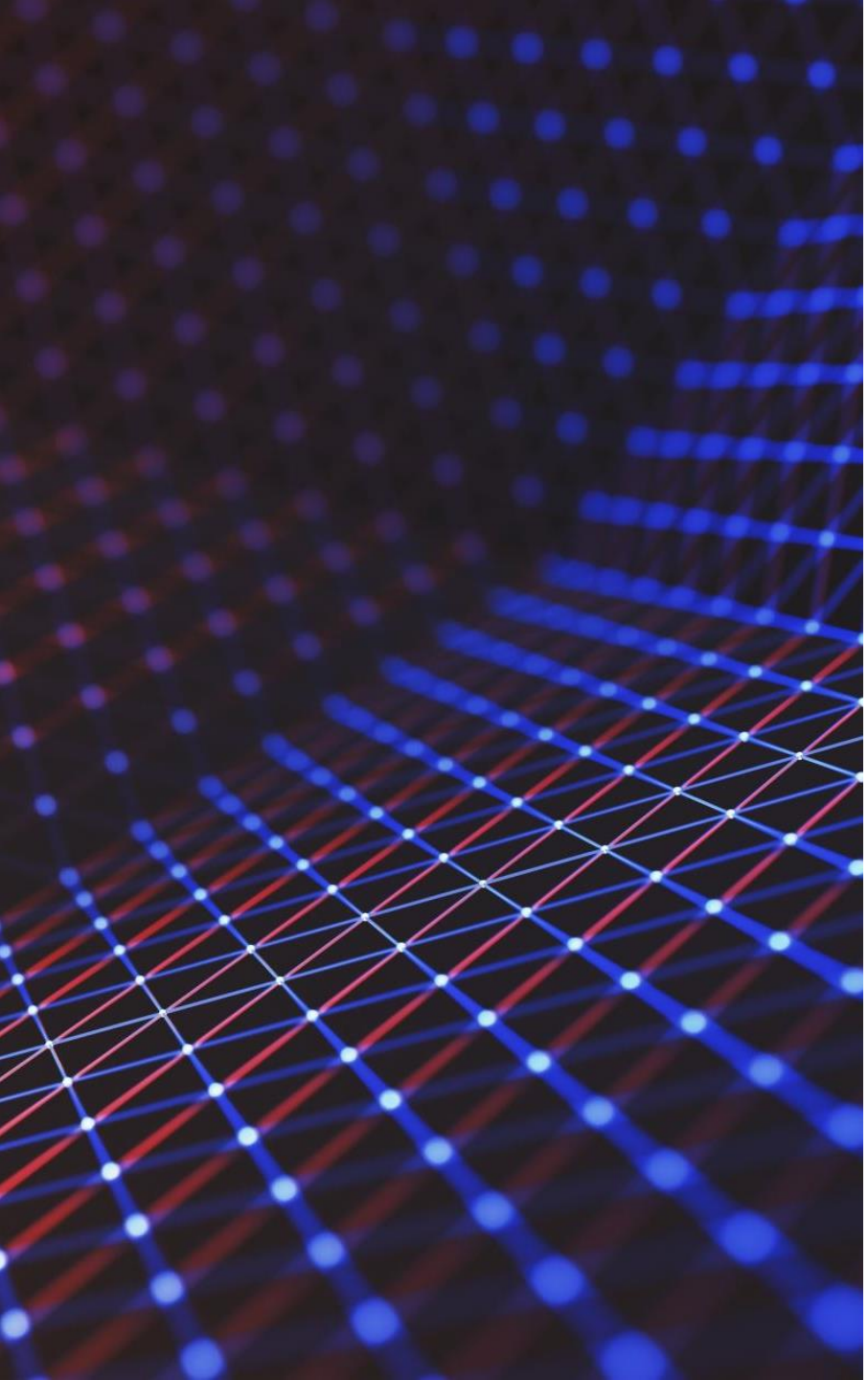
Universal Search

Document AI



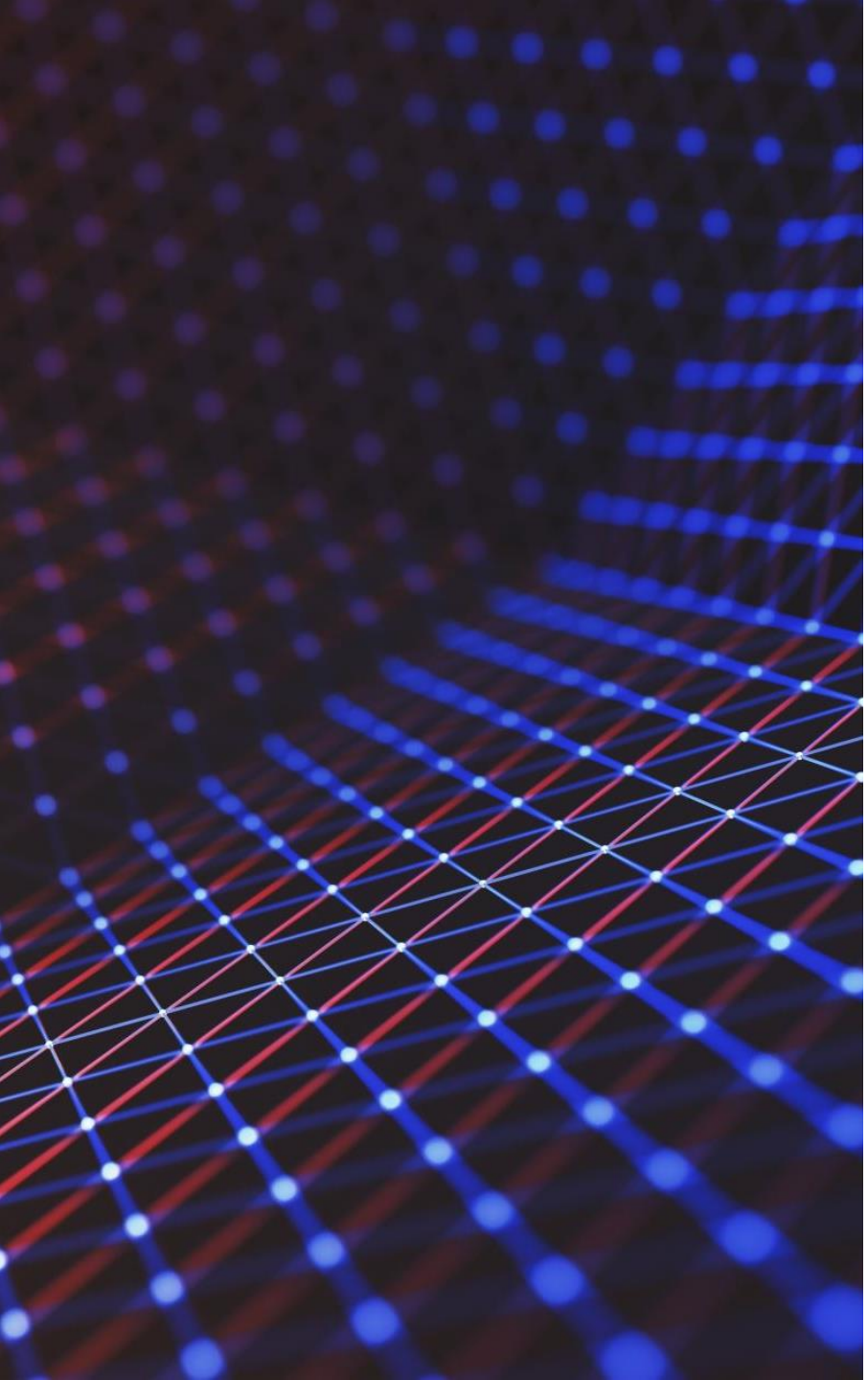
LABELING DATA

- f(X) = y → **model(features) = labels**
- **model** = f, *function* (? → ML)
- **feature** = column in a tabular dataset besides the label column; X, input (day of week, weather), numeric repr of an aspect of raw data
- **label** = y, output (# of rentals for that day)
  - ***labeled data*** = w/ single/multiple target attributes/columns (dep vars)
  - ***unlabeled data*** = no target attribute/label



ML PROBLEM IDENTIFICATION

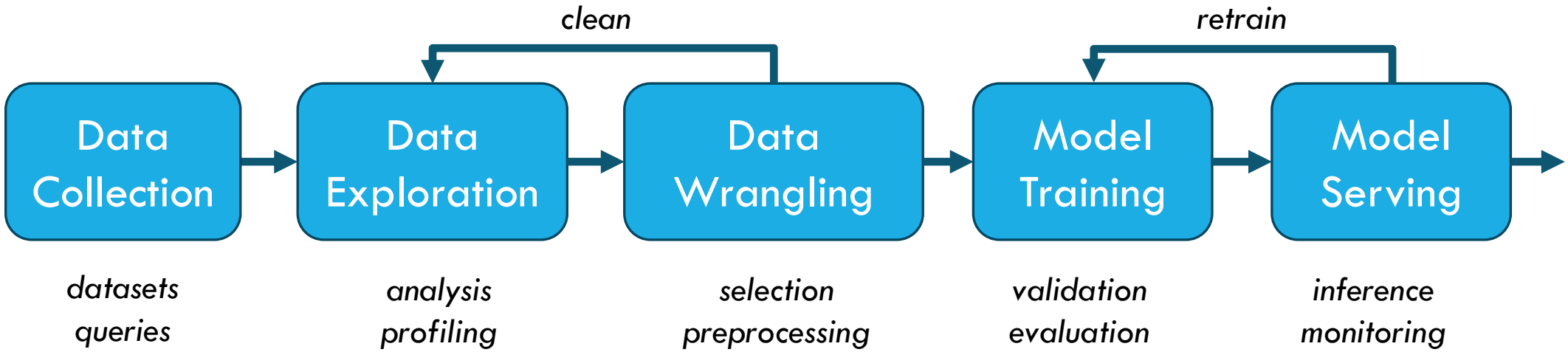
- *problem types*: regression/classification, clustering…
- **statistics:** f is known → apply for unknown values
  - linear/logistic regression
  - correlation
  - distributions
- **ML:** f(features) → labels – model training/inference
- **DL:** polynomial fct

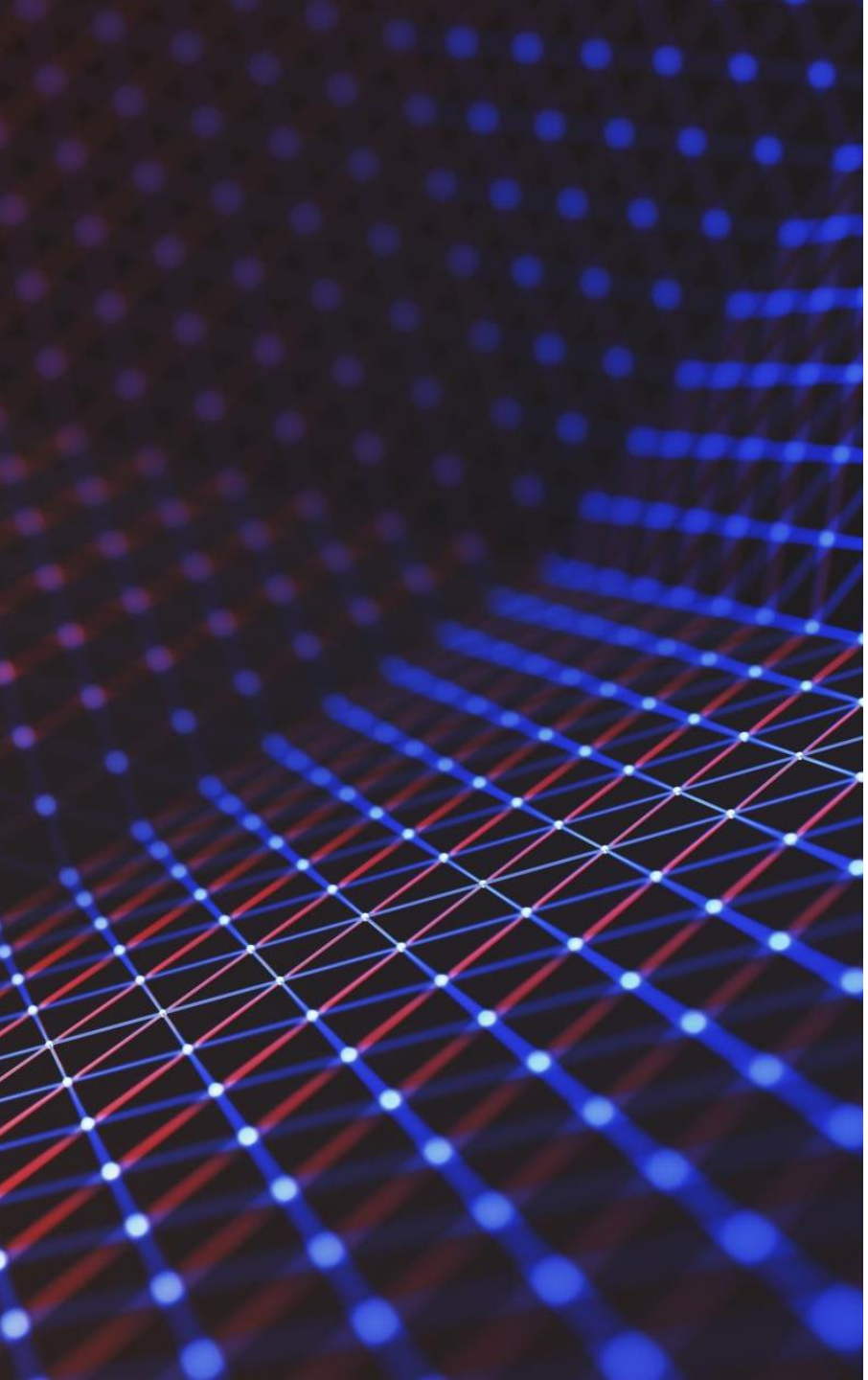


MACHINE LEARNING PARADIGMS

- **Supervised Learning** = w/ labeled data, to train models
- **Semi-Supervised (Weak) Learning** = w/ one portion labeled
- **Unsupervised Learning** = learn patterns from unlabeled data, w/ no human supervision
- **Reinforcement Learning** = choose actions to maximize rewards
- **Transfer Learning** = enhance previously trained models
- **Ensemble Learning** = w/ multiple algs (weak learners) → final prediction after majority vote

MACHINE LEARNING PIPELINES

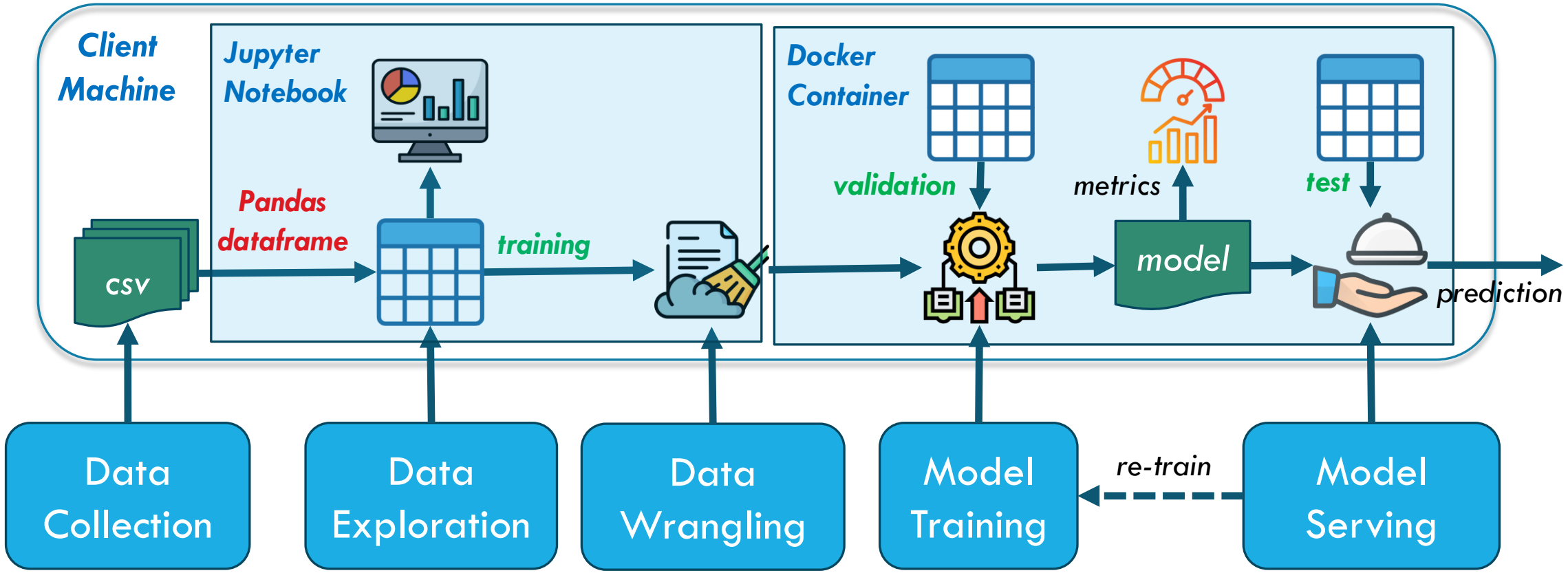




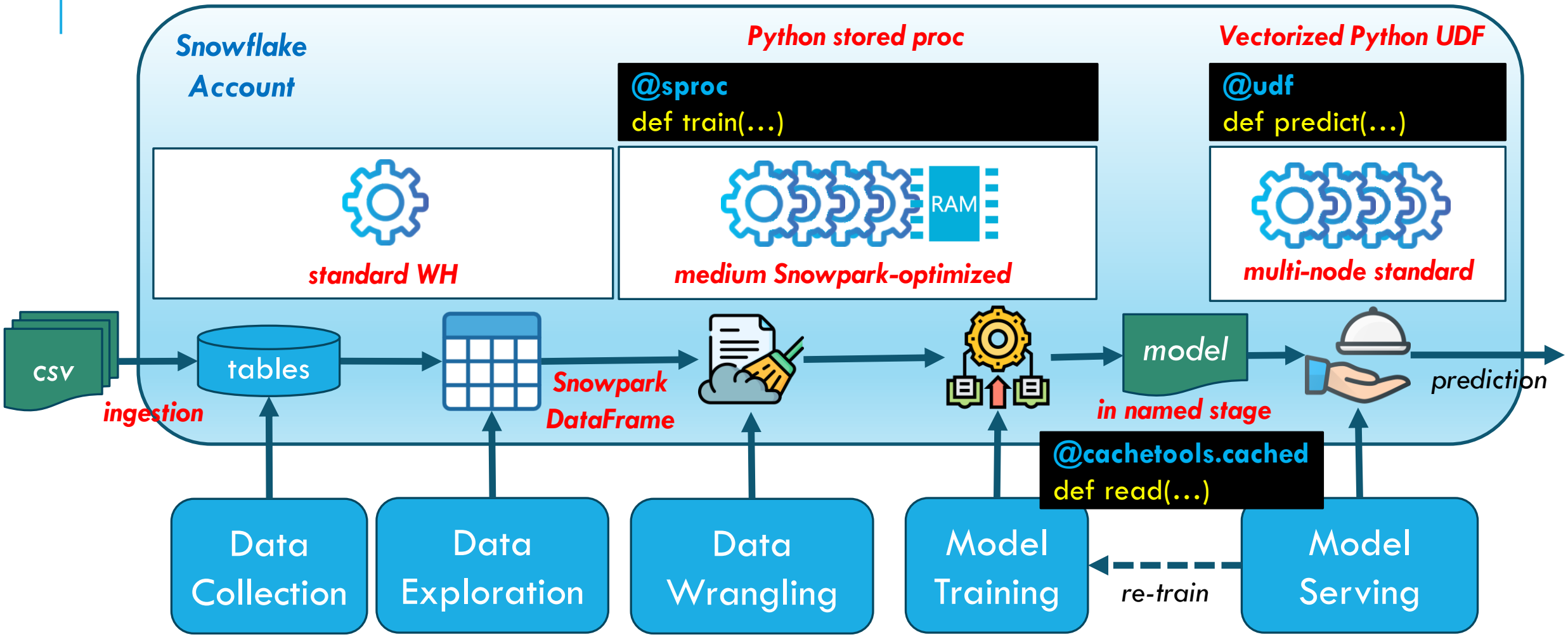
MAIN ML PIPELINE STEPS

- **Data Collection** = find and prepare raw datasets (CSV, Parquet - in data lakes) + upload/ingest data into tabular format (in tables)
- **Data Exploration** = data analysis, to discover and understand data (+correlations), using data profiling and data visualizations
- **Data Wrangling** = feature selection/extraction + cleaning (impute missing values...) + preprocessing (normalization/encoding...)
- **Model Training** = fit model to training data, evaluate (performance metrics) + validate (CV)
- **Model Serving** = persist trained model (in a registry) + reuse to serve/predict (w/ auto-scale), monitor model drifting...

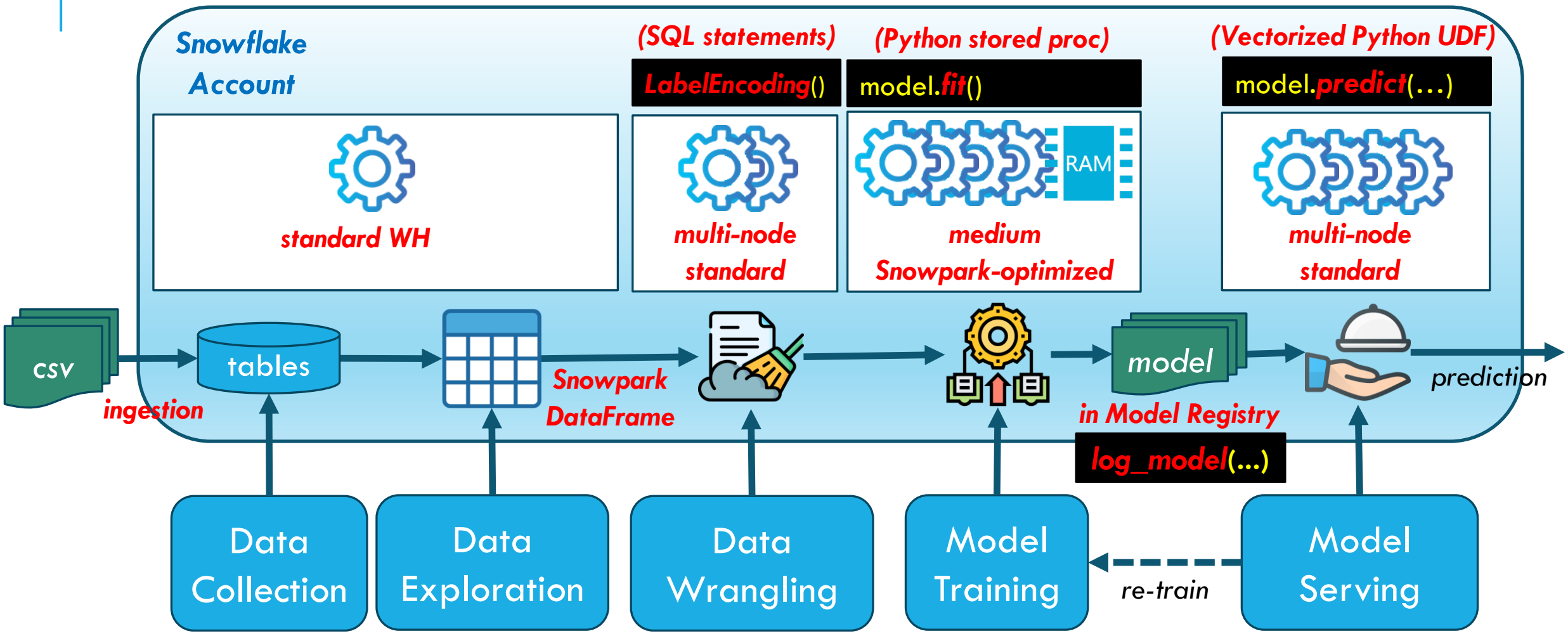
ML PIPELINES ON DATASETS (OUTSIDE SNOWFLAKE)



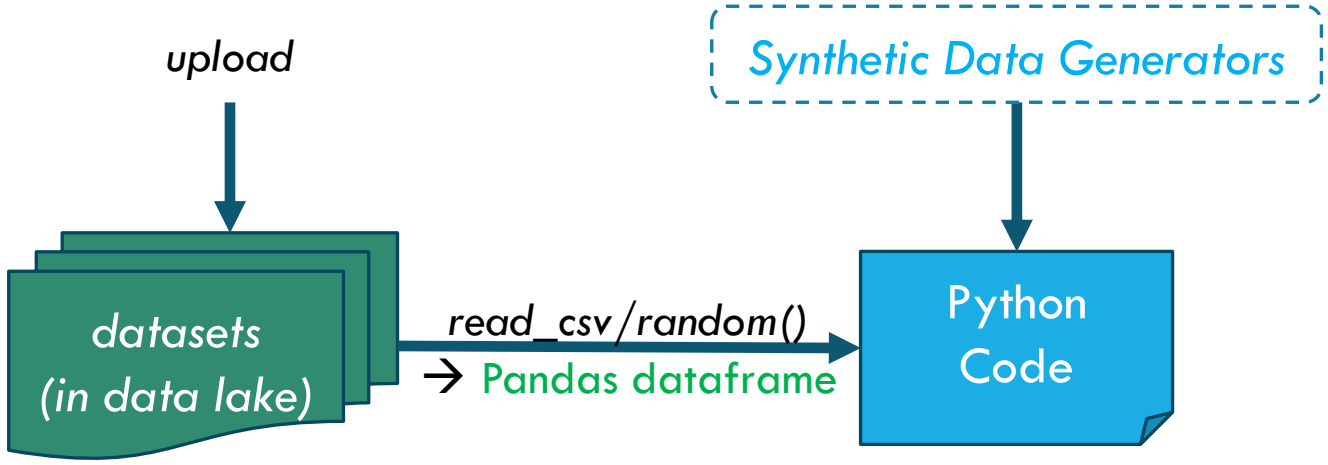
ML PIPELINES USING SNOWPARK (NO CORTEX)

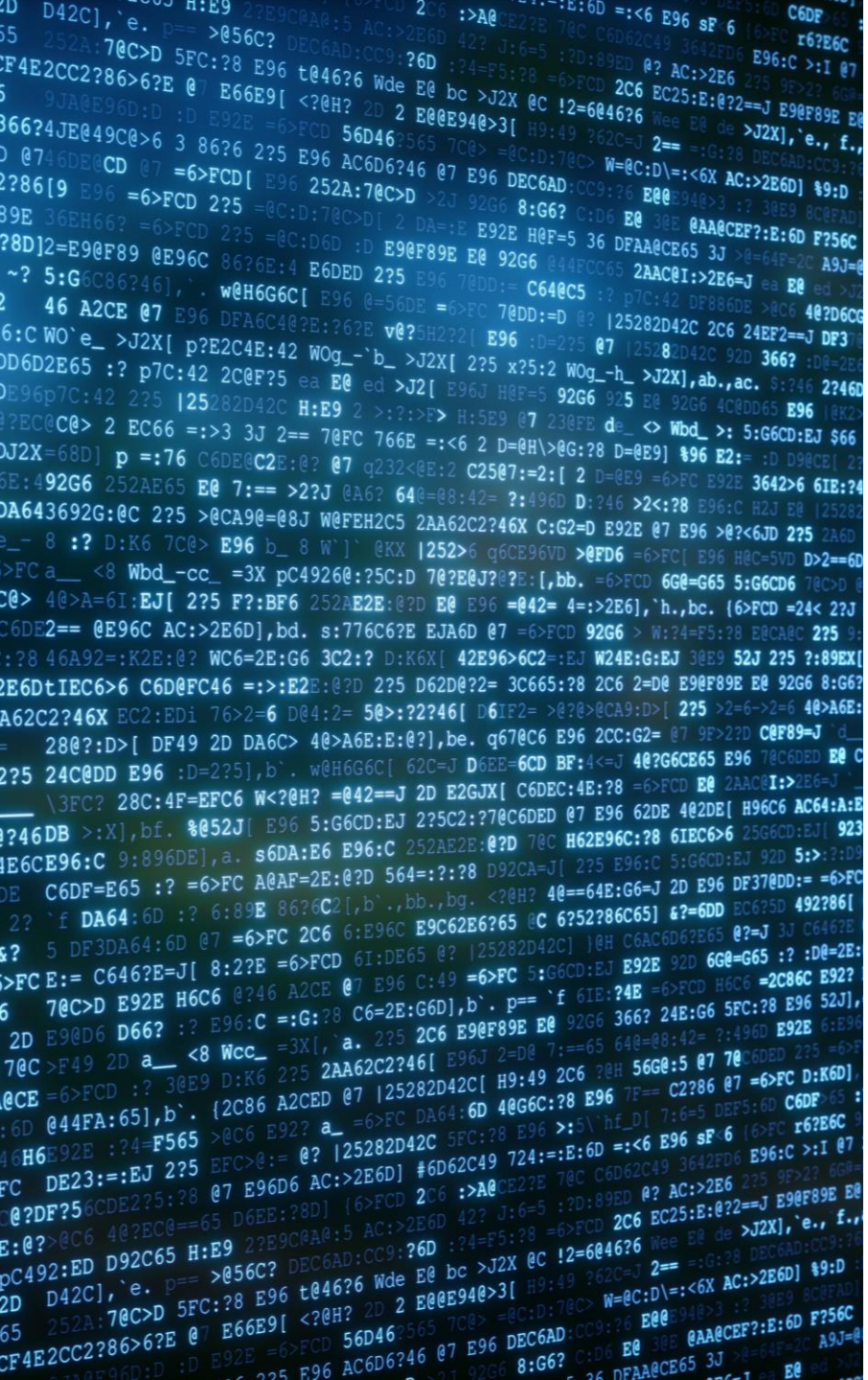


ML PIPELINES WITH SNOWPARK ML (IN CORTEX)



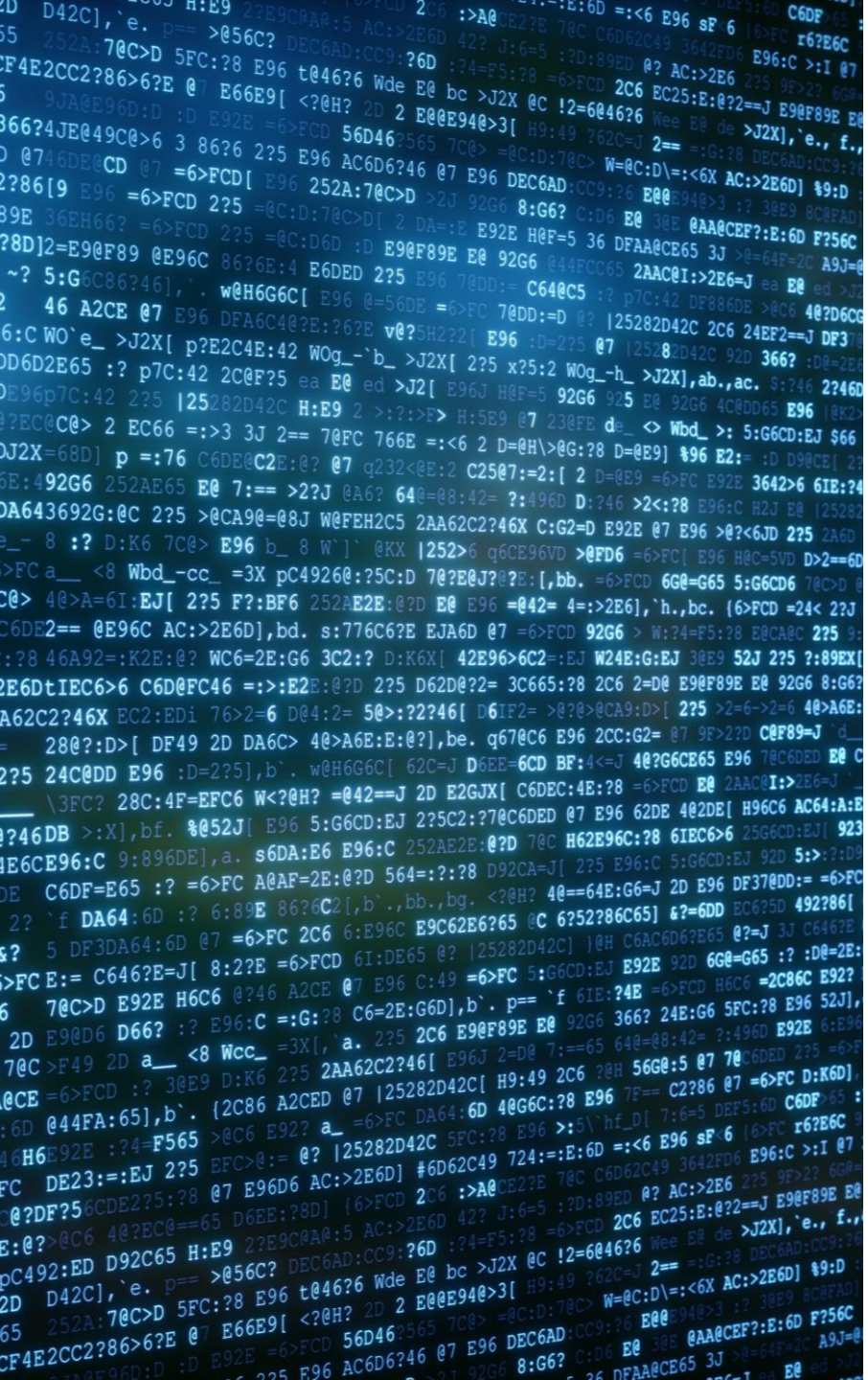
DATA COLLECTION (FROM DATA LAKE)





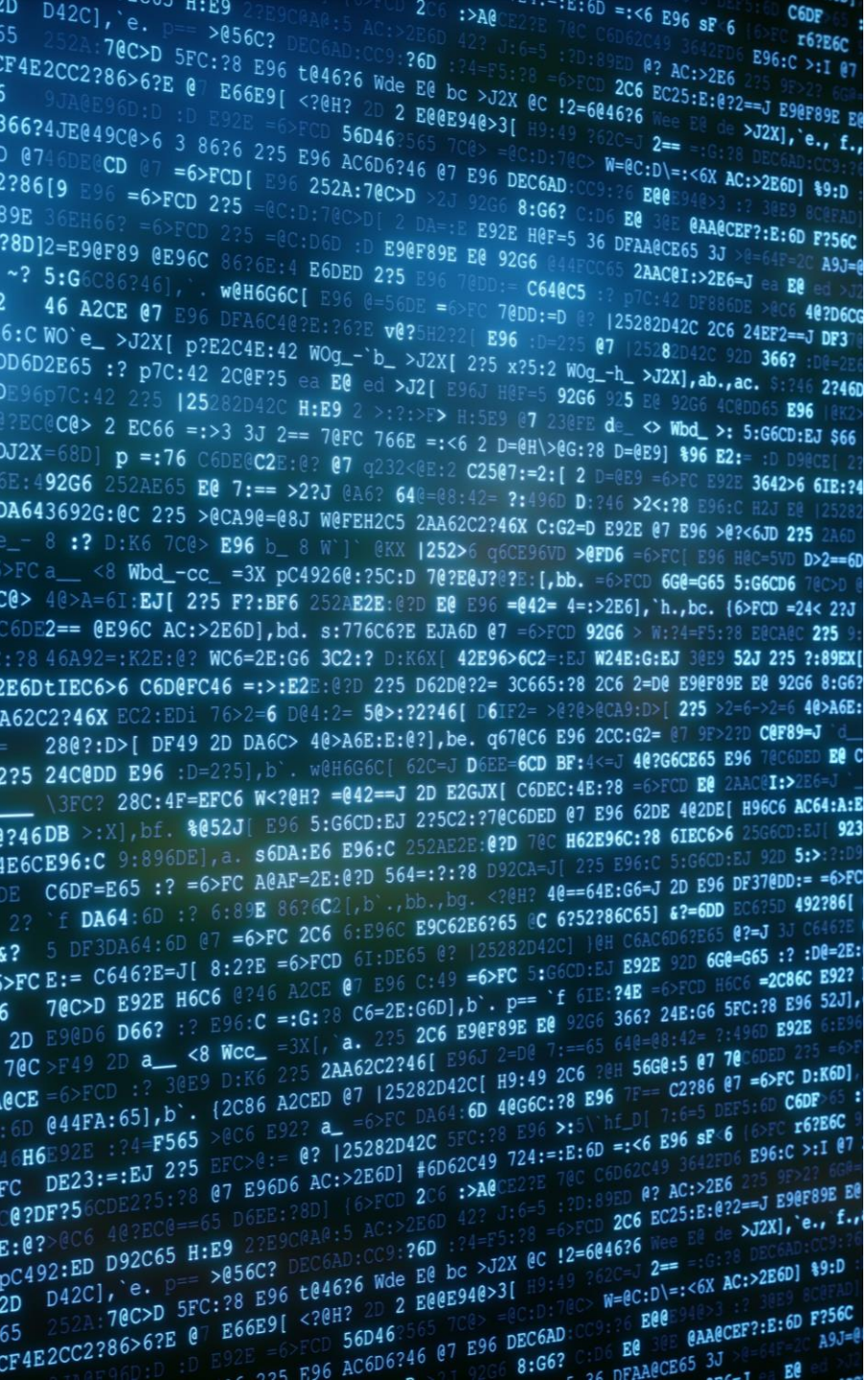
PUBLIC DATASETS

- AWS public datasets (440)
- FiveThirtyEight datasets
- Kaggle datasets
- Google BigQuery public datasets
- Wikipedia database download
- NASA Earth data
- US Census
- **Snowflake QuickStarts**



FRAMEWORK BUILTIN DATASETS

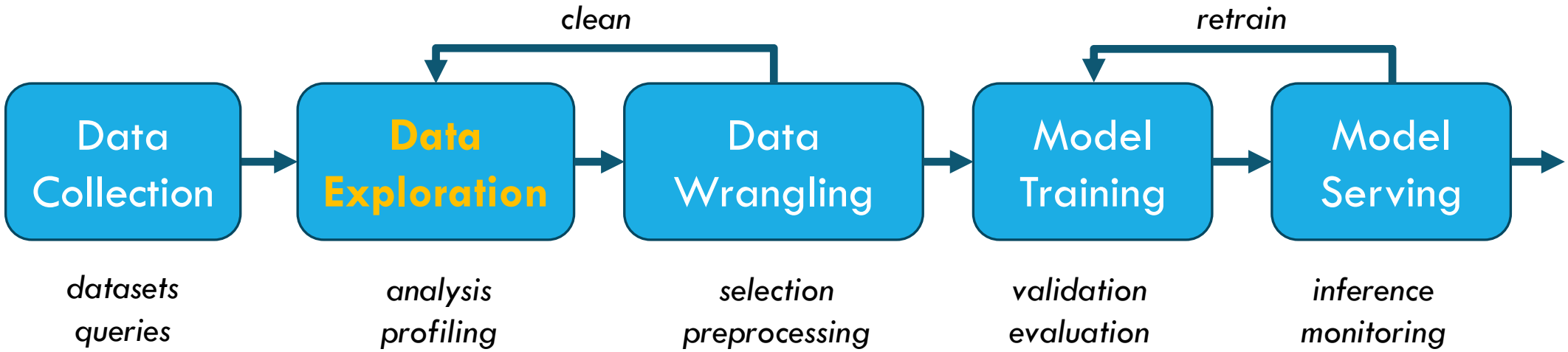
- sklearn toy datasets ← ***sklearn.datasets***
- Keras built-in datasets ← ***tf.keras.datasets***
- TensorFlow datasets ← ***tf.data.Datasets***
- PyTorch built-in datasets ← ***torchvision.datasets***

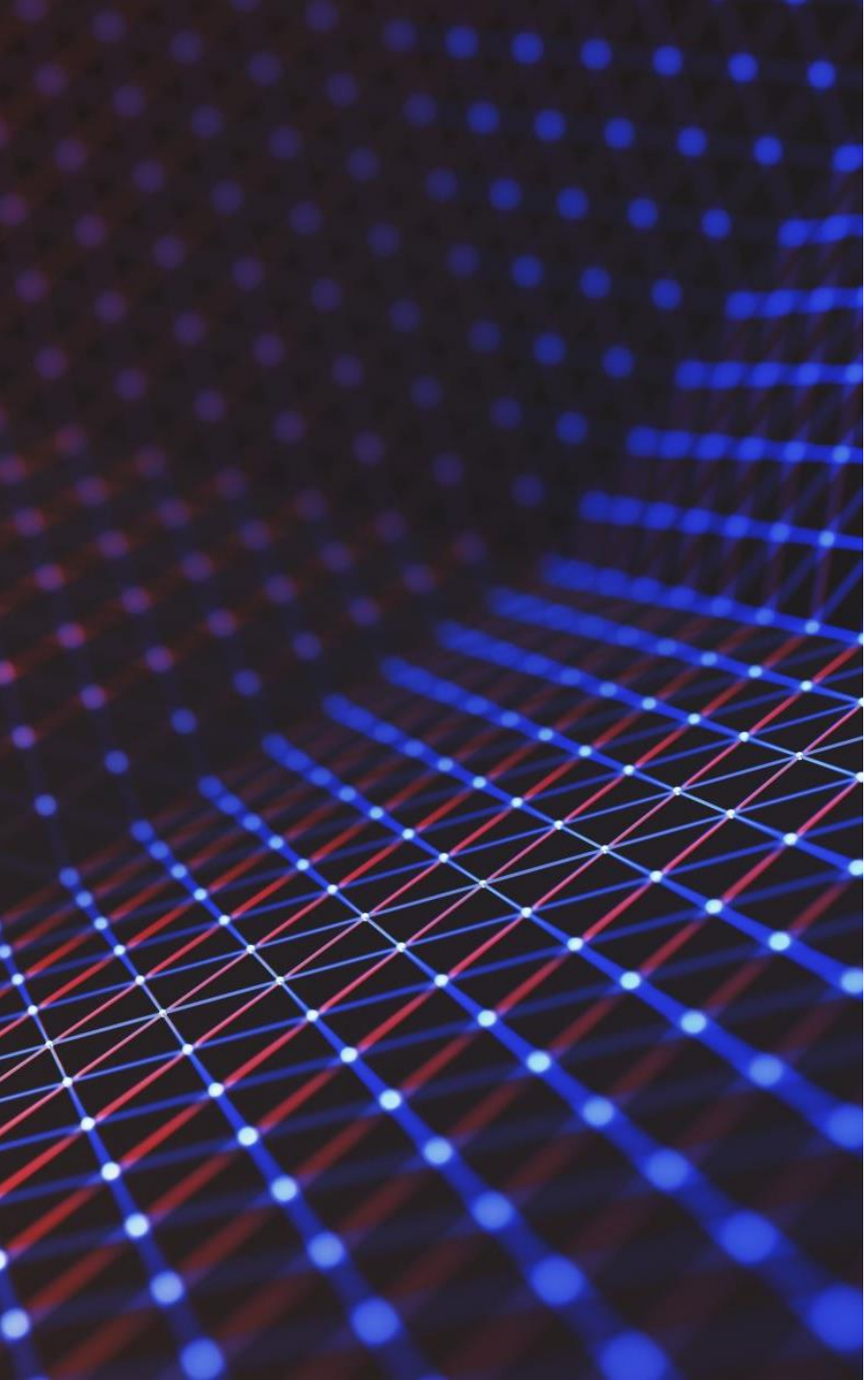


SKLEARN TOY DATASETS

- *load_iris*(...) → iris dataset - for classification, 3 classes, 50 samples/class (150 total)
- *load_breast_cancer*(...) → breast cancer Wisconsin dataset - for classification, 569 instances
- *load_diabetes*(...) → diabetes dataset - for regression, 442 instances, first 10 columns numeric predictive
- *load_digits*(...) → digits dataset - for classification, 1797 instances, 64 attributes, 8x8 pixel images w/ 0..16
- *load_wine*(...) → wine dataset - for classification, class_0/1/2 (3 classes): [59,71,48] (=178 total samples)
- *load_linnerud*(...) → physical exercise Linnerud dataset - 20 instances, 3 attributes

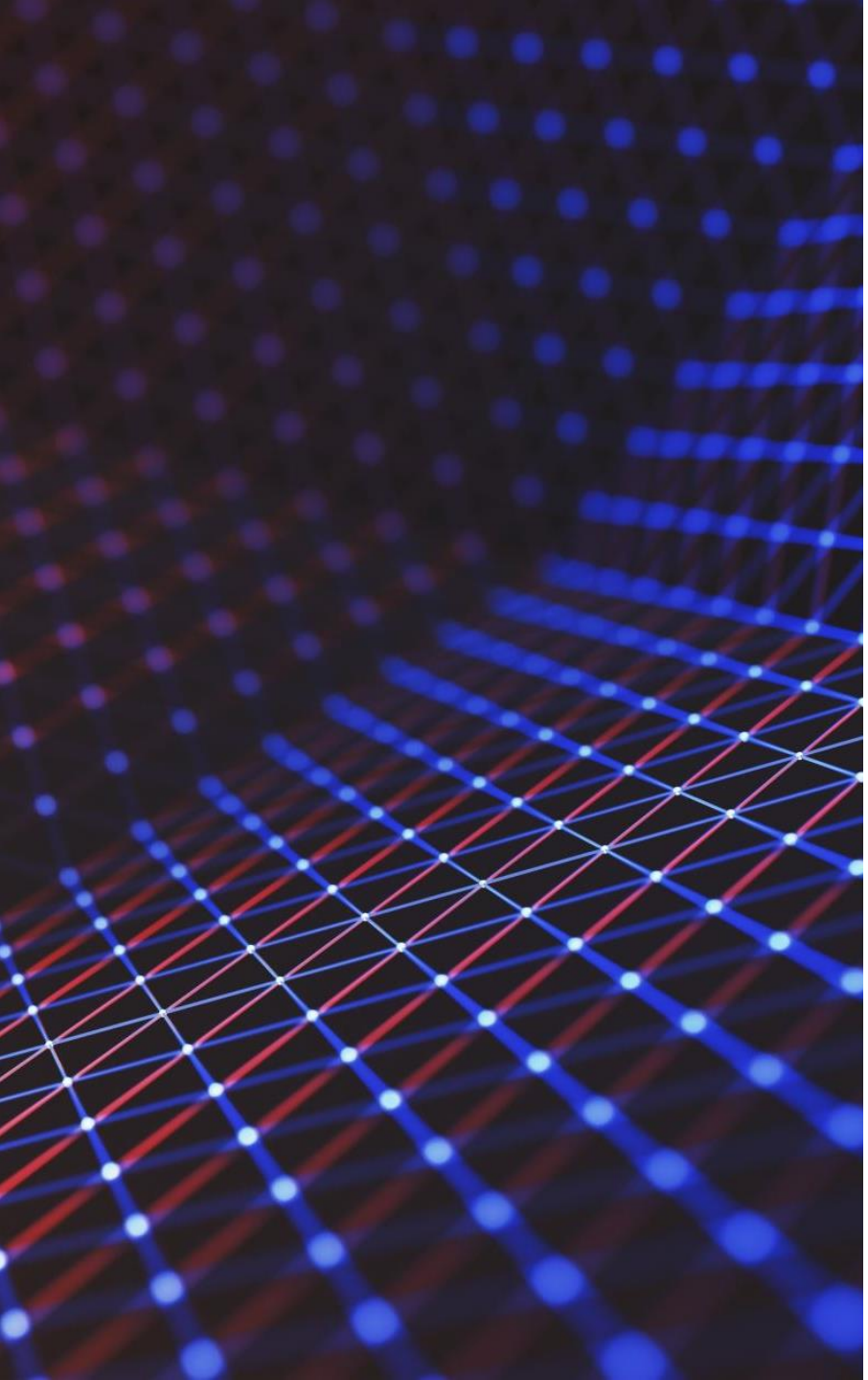
MACHINE LEARNING PIPELINES





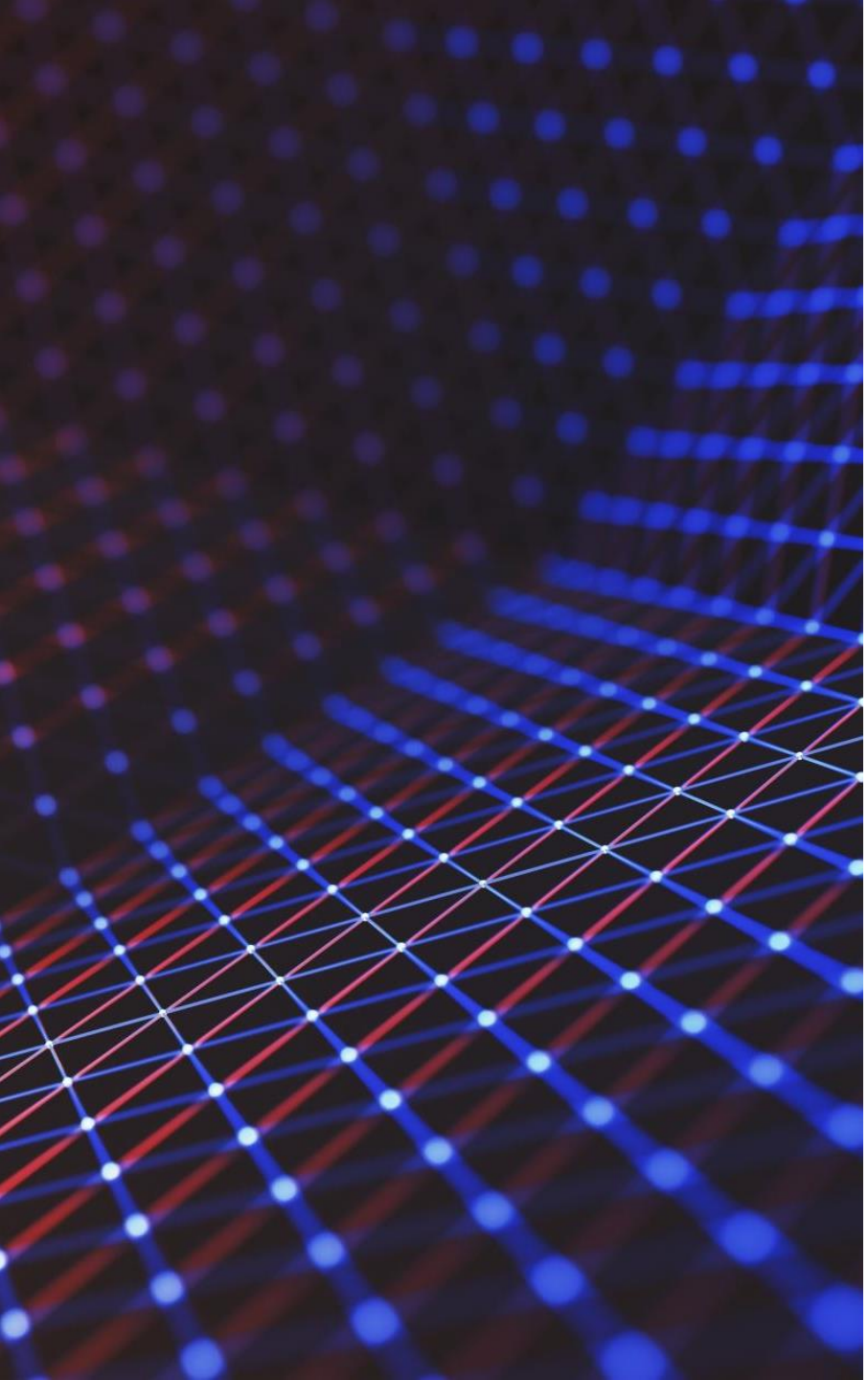
DATA EXPLORATION

- **EDA (Exploratory Data Analysis)** = to identify general patterns (outliers, unexpected features)
- **AHA (Ad-Hoc Analysis)** = to find quick answers to a single immediate question
- **Data Profiling** = examine/analyze/create useful summaries of data (min/max/mean, distinct/missing, value distribution...) → to discover data quality issues, risks, overall trends
- **Data Visualization** = communicate complex data relationships and data-driven insights in a way easy to understand (charts, plots, infographics, animations...)
- **Data Correlation** = how two variables/features are linearly related → ***correlation heatmap/matrix***



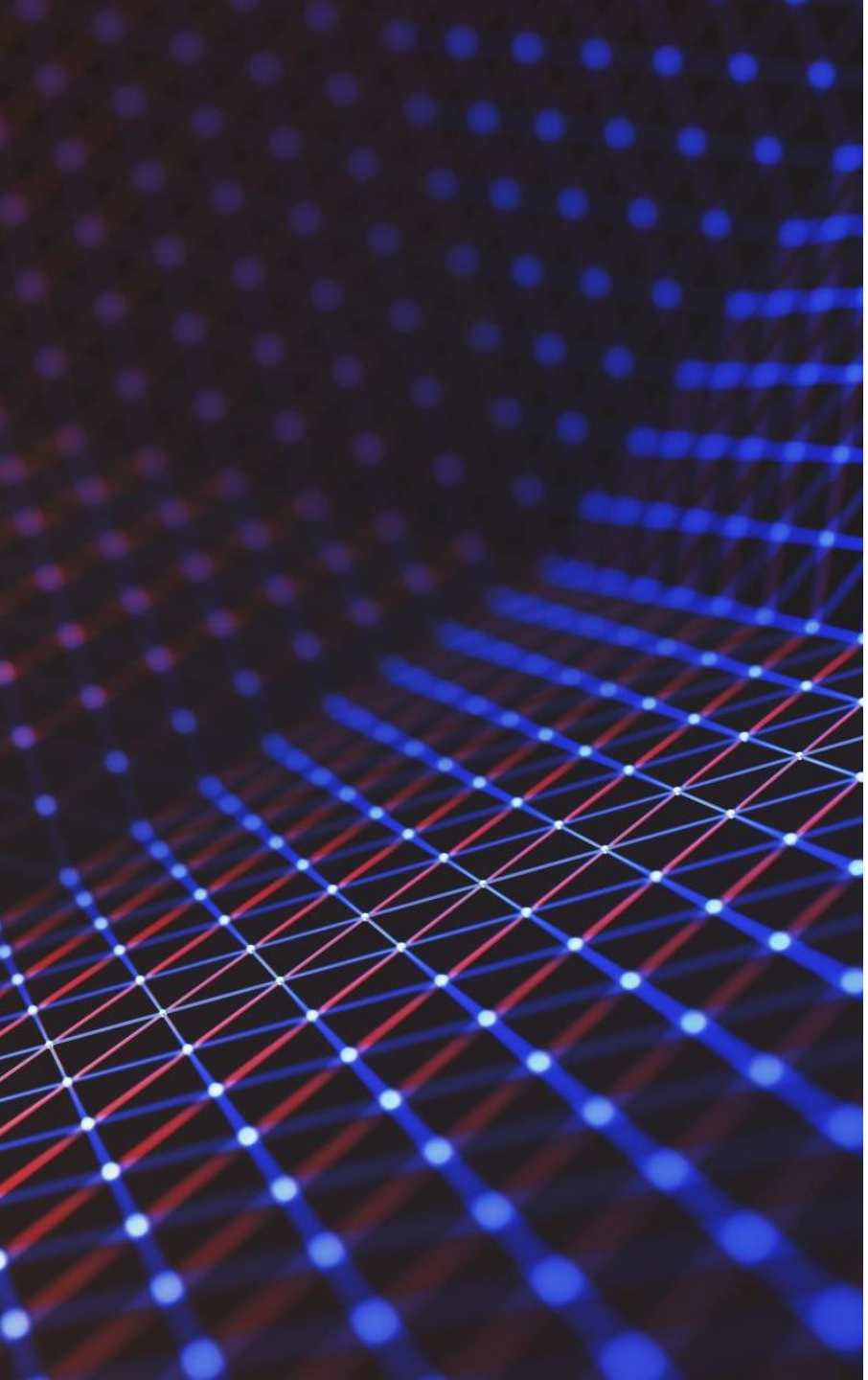
DATA ANALYSIS

- **Descriptive** = what has happened (w/ historical data): summarize large datasets w/ KPI
- **Diagnostic** = why it happened (find reason/cause on historical data): identify anomalies → collect related data → discover relships/trends w/ statistics
- **Predictive** = what will happen (in the future): historical data → trends – w/ statistics/ML (CNN, decision trees, regression)
- **Prescriptive** = what to do (to achieve a goal/target): ML → patterns in large datasets
- **Cognitive** = draw inference from existing data/patterns: in ML/NLP



DATA PROFILING

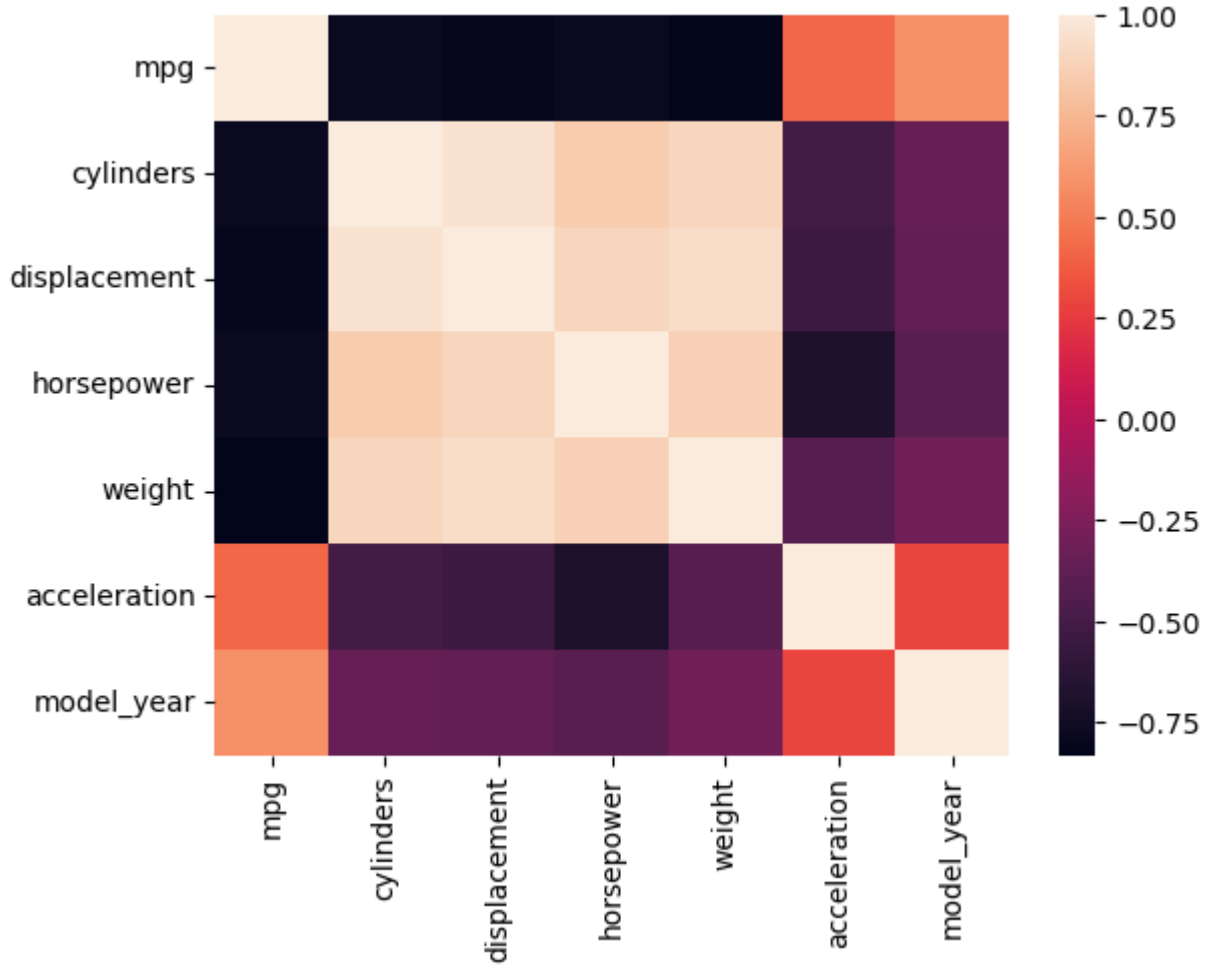
- top/bottom 10 sample values
- data types → numerical+categorical variables
- min...max + mean/sum values
- missing cells (NULL values)
- total count + size
- total distinct+duplicate values
- total zero/positive/negative values
- avg string length + discrete text values → WordCloud
- distribution → 25/50/75% quantile statistics
- standard deviation + kurtosis/skewness



DATA VISUALIZATION

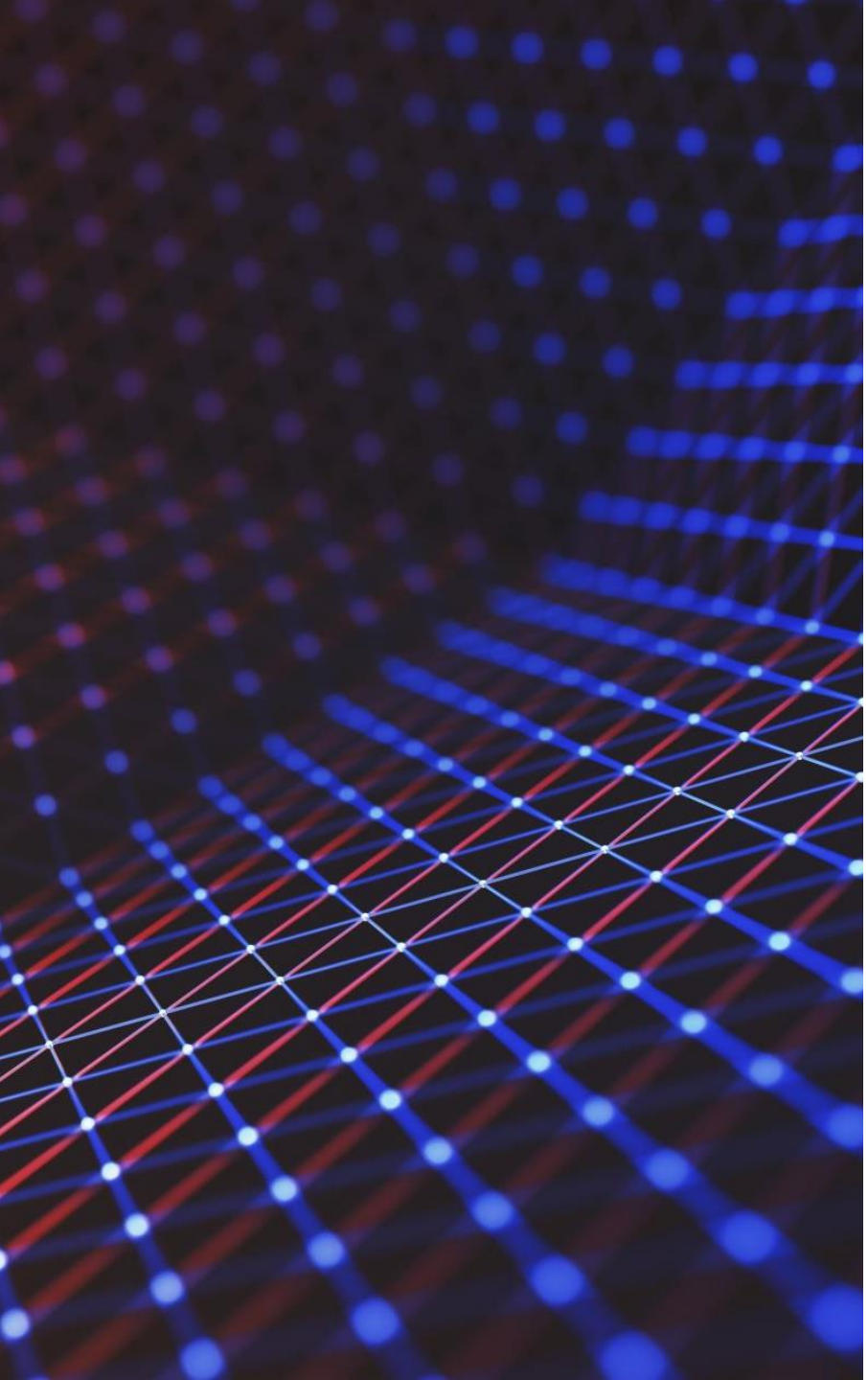
- ***chart types***
  - scatter plot, box plot
  - histograms/density → for data distributions
  - heatmaps → for correlations
  - line/bar charts
- ***dataviz libraries & products***
  - matplotlib, Seaborn
  - Plotly, Altair
  - Pillow(PIL) → for image rendering
  - Tableau / Power BI/ Looker / Excel

CORRELATION HEATMAP/MATRIX



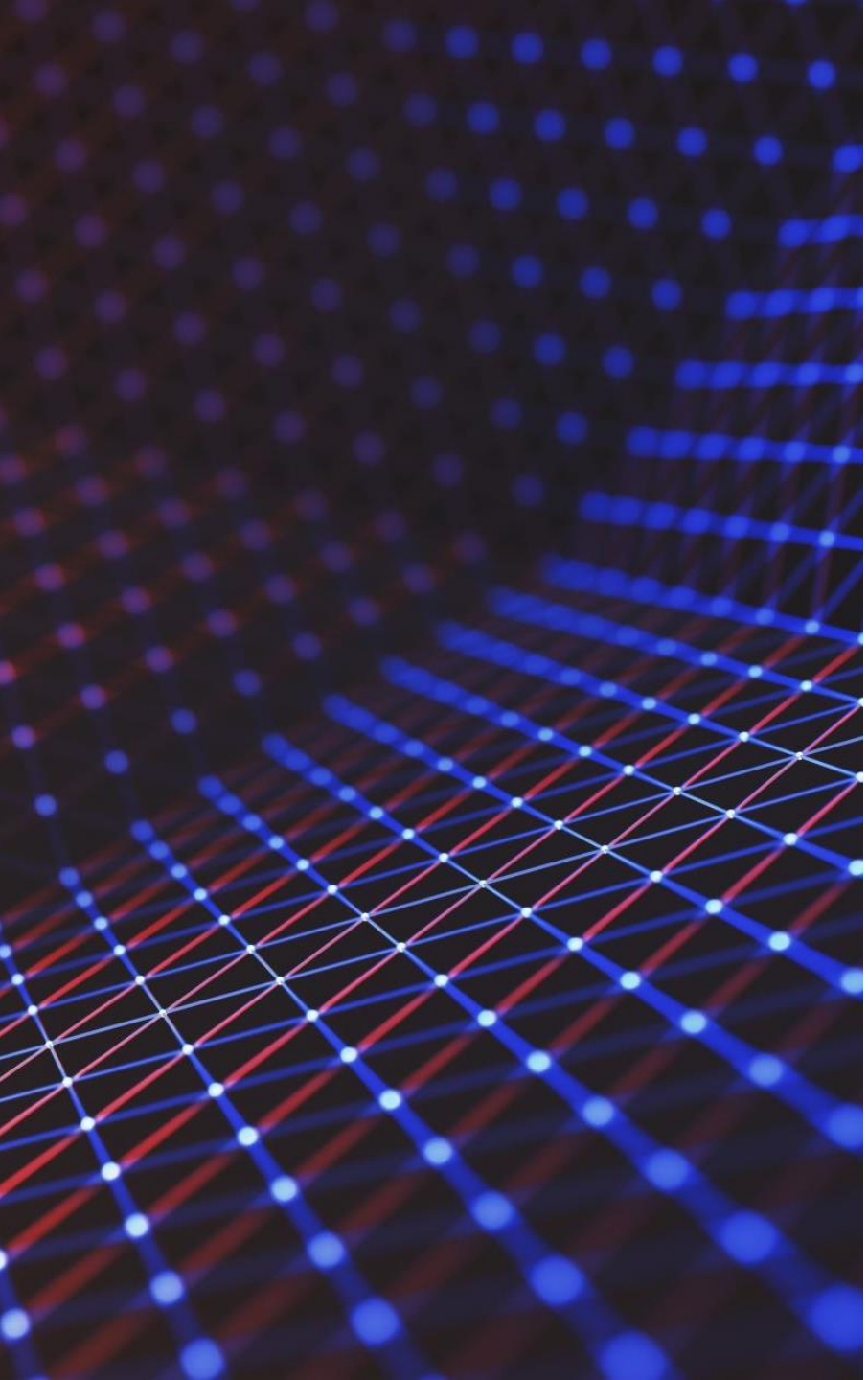
DATA WRANGLING

- = process to convert/transform raw data into better structures/formats
- **Data Discovery** = to understand data
  - data analysis/exploration/visualization
  - data profiling → min/max/mean/mode, cardinality
- **Data Structuring** = in relationships…
- **Data Cleaning** = same format, remove outliers…
- **Data Enriching** = need to add additional data?
- **Data Validation** = for data consistency/quality
- **Data Publishing** = for use downstream



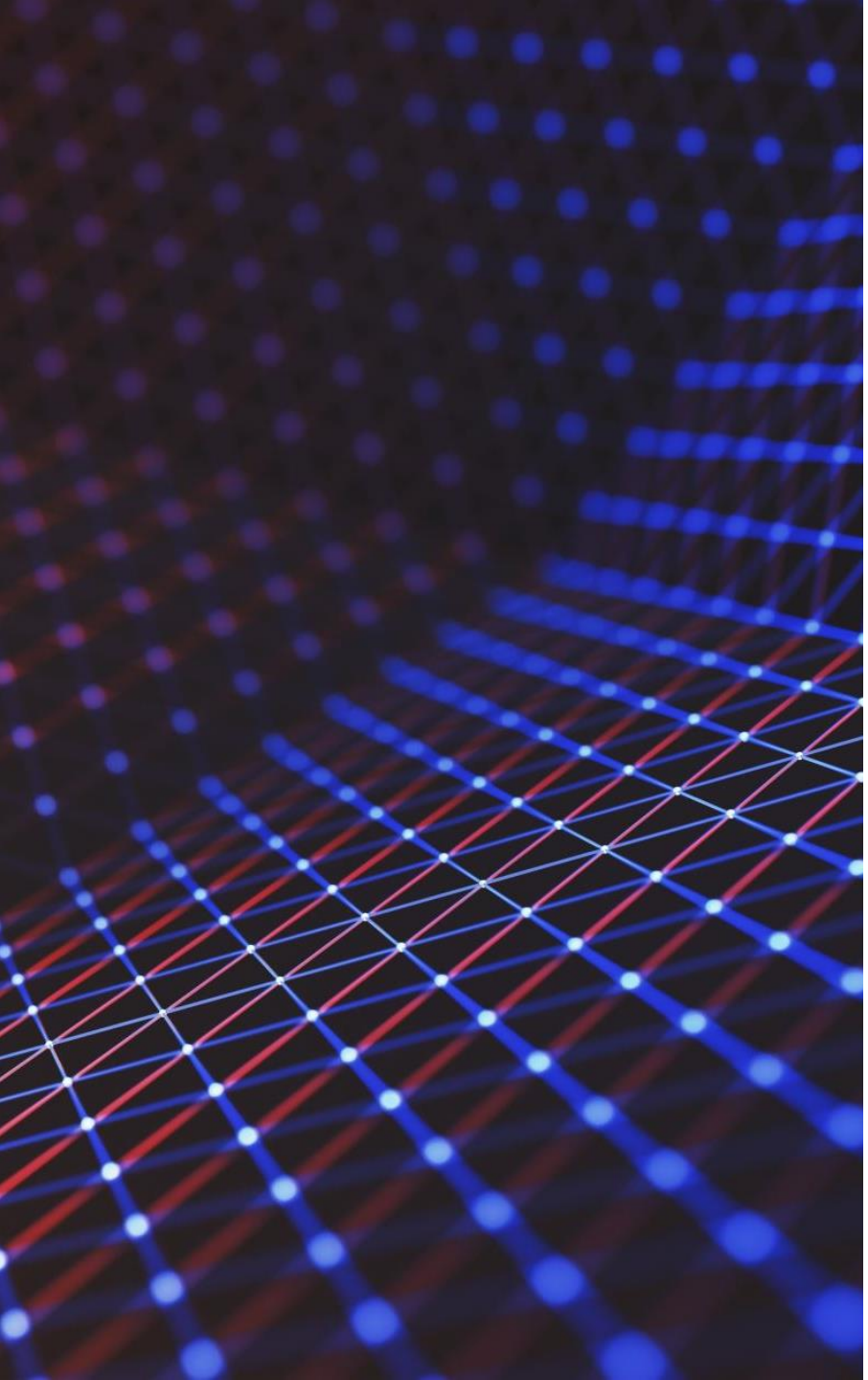
FEATURE ENGINEERING

- **Feature Improvement** = make existing features more usable → **math transforms / impute missing data**
- **Feature Construction** = **data augmentation / derive new features**
- **Feature Selection** = choose best subset from existing set of features → remove features / combine features / dep&indep features that cancel each other / synthetic features
- **Feature Extraction** = rely on algs to create new features / transfer learning / compose features (PCA = selection + extraction)
- **Feature Learning** = generative AI → auto-gen new set of features



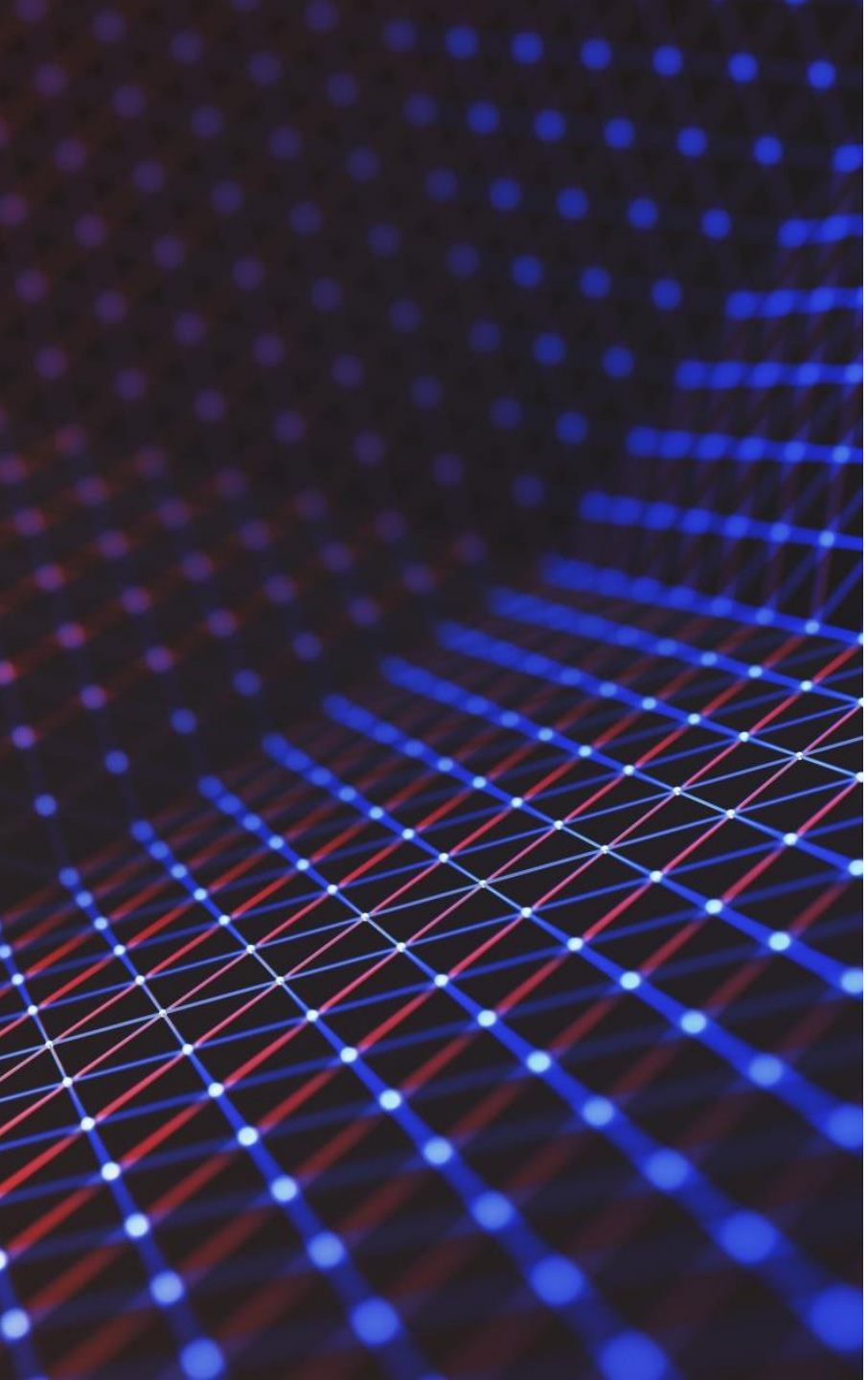
FEATURE TRANSFORMATIONS

- *Function Transformer*
  - **Log Transform** → if right-swkewed
  - **Square Transform**
  - **Square Root Transform** → if left-swkewed
  - **Reciprocal Transform**
  - **Custom Transform**
- *Power Transformer*
  - **Box-Cox transform**
  - **Yeo-Johnson transform**
- *Quantile Transformer*
  - output_distribution=uniform/normal



IMPUTATION TECHNIQUES

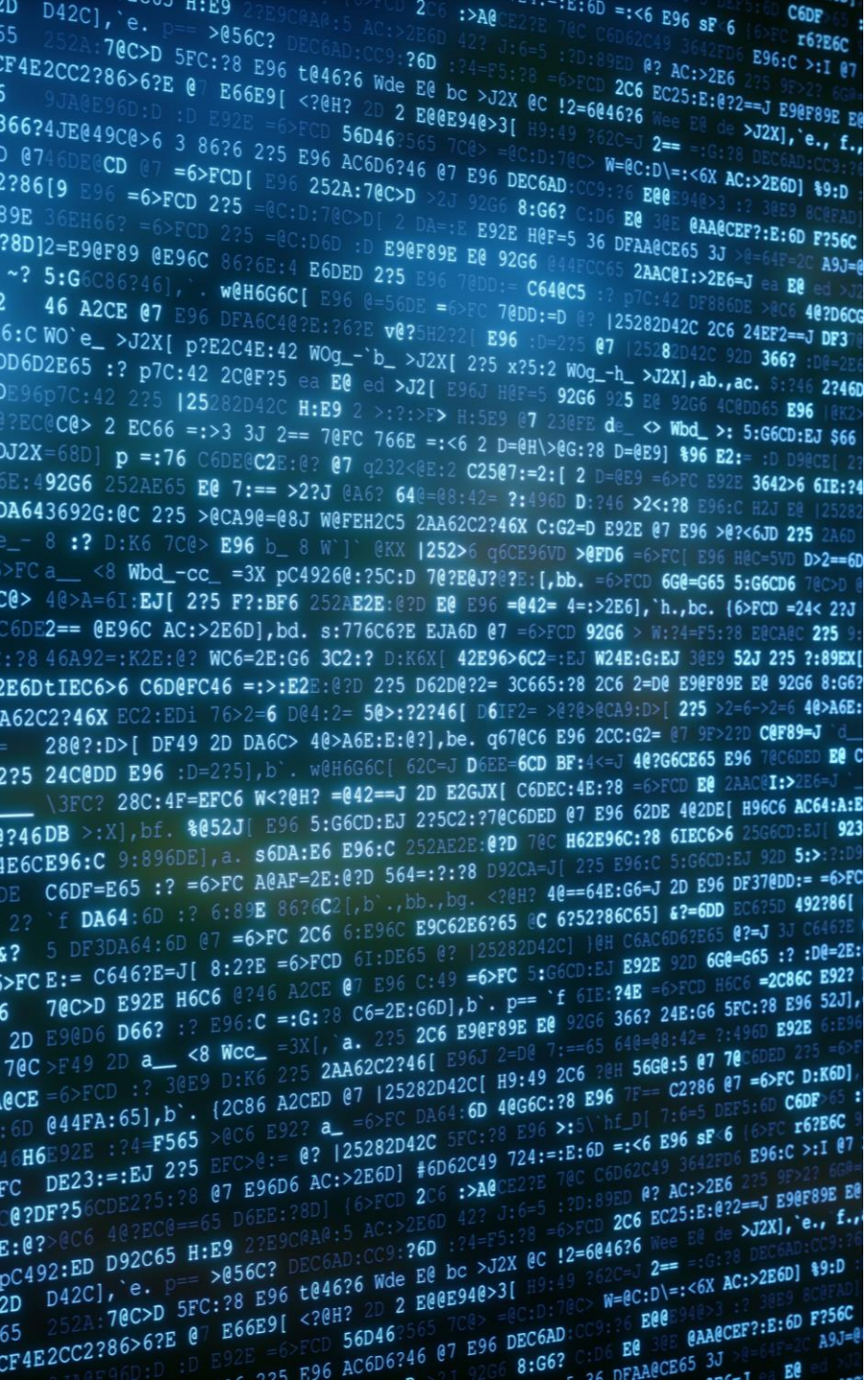
- **SimpleImputer** → **Zero/Mean Imputation** = replace missing values by 0/mean
- **KNNImputer** → **kNN-Imputation** =
- **IterativeImputer** → **Iterative Imputation** = round-robin linear regression



TRANSFORMERS

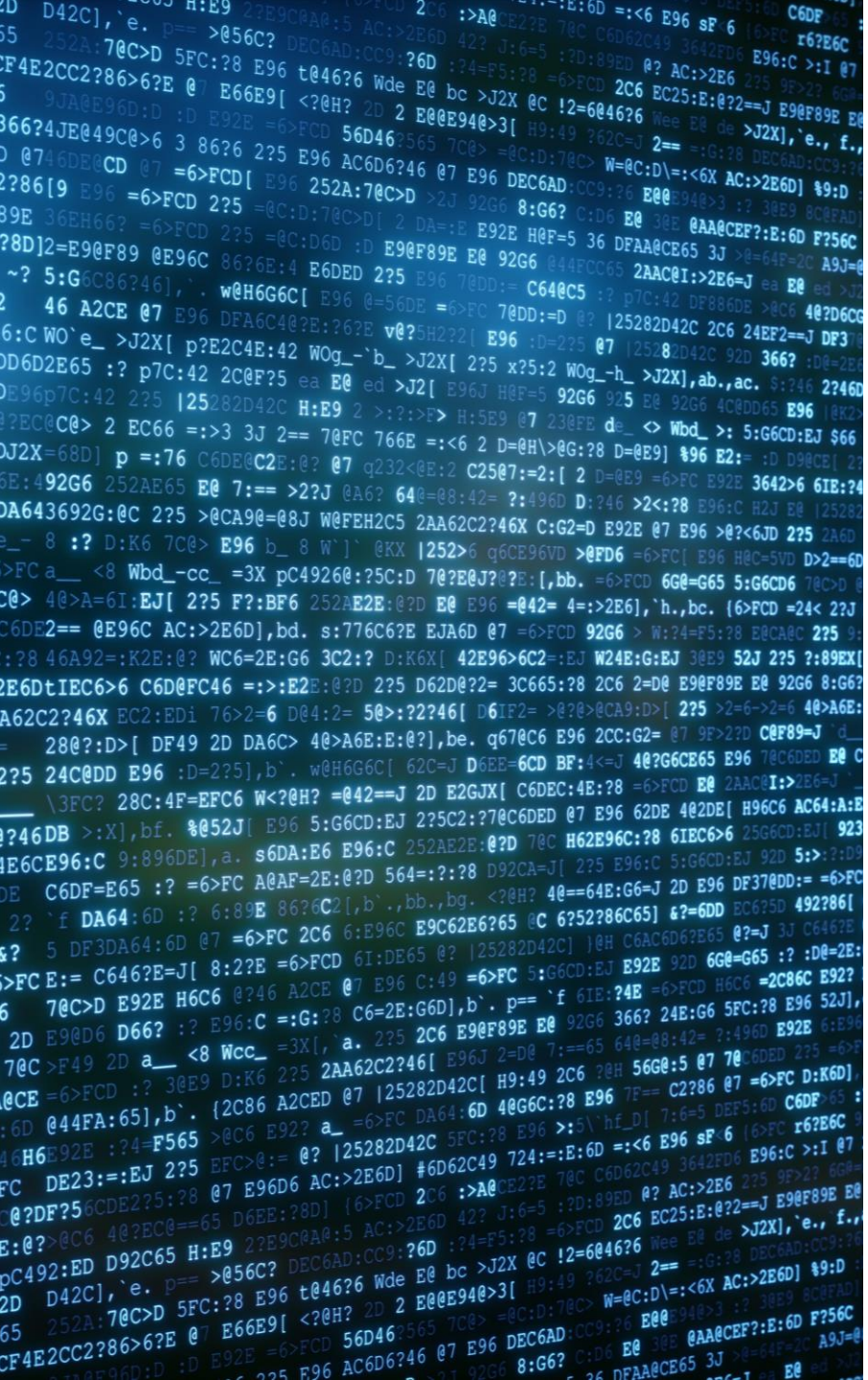
- *normalizers* → **Normalizer**
- *scalers* → **Standard/MinMax/MaxAbs/Robust**
- *encoders* → **OneHot/Ordinal/Label**
- *binarizers* → **Binarizer, KBinsDiscretizer**

- **ColumnTransformer**
- **Pipeline**



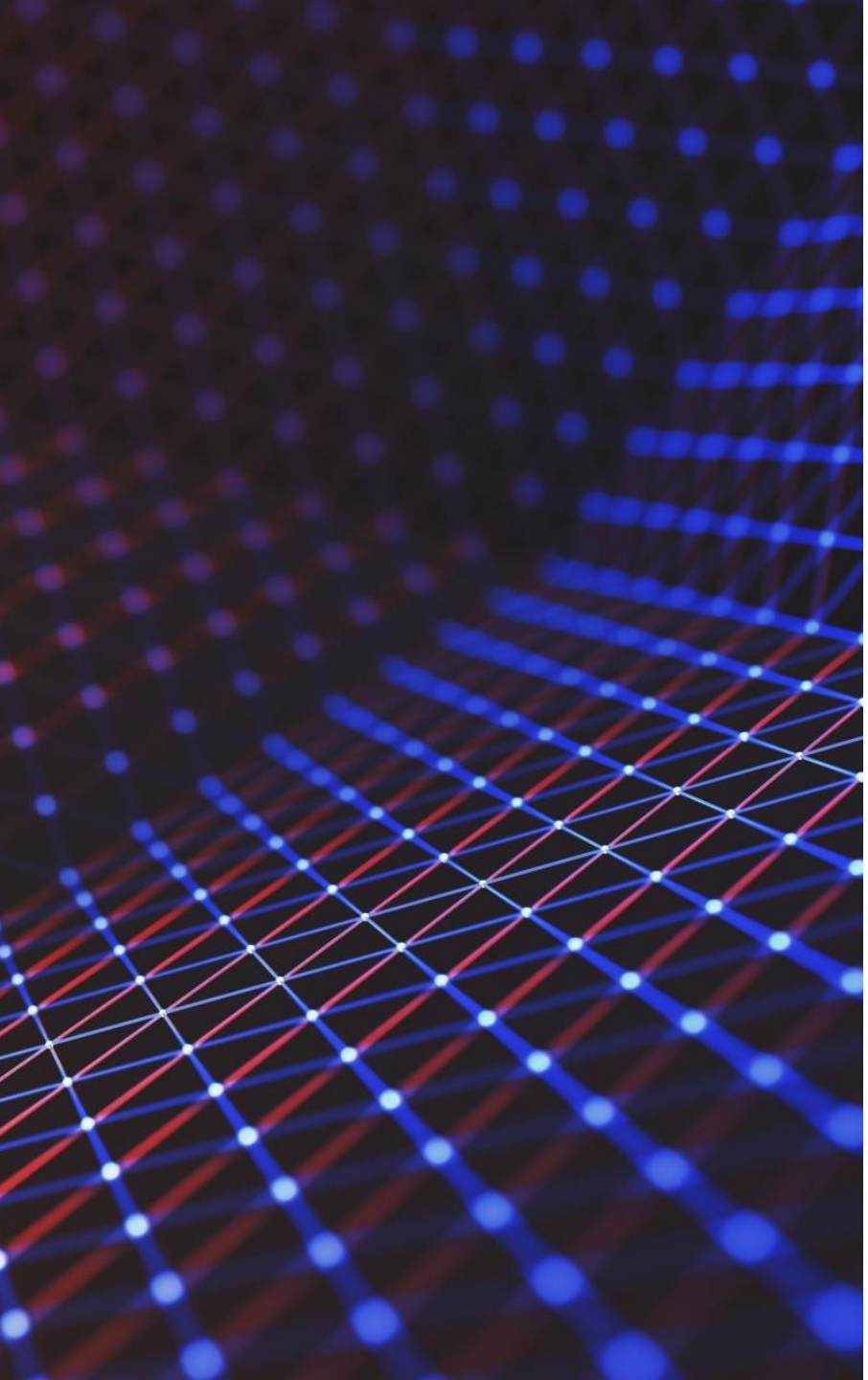
MODEL TRAINING

- **Data Preprocessing** = can be part of the same pipeline
- **Data Segregation** = into train/validate/test subsets
- **Model Training** = w/ fit/transform → predict
- **Model Validation** = w/ Cross-Validation, ex: KFold
- **Model Evaluation** = from Performance Metrics, ex: AUC
- **Hyperparameter Optimization** = ex: GridSearch



DATA SEGREGATION

- = ***train*** → [***validate*** → tune hyperparams] → ***test***
- **training data** = map input → predicted output ← to learn parameters of the model
- **validation data** = measure quality of model predictions ← to tune hyperparameters
- **test data** = to which model has never been exposed, after hyperparameters tuned ← w/ predict

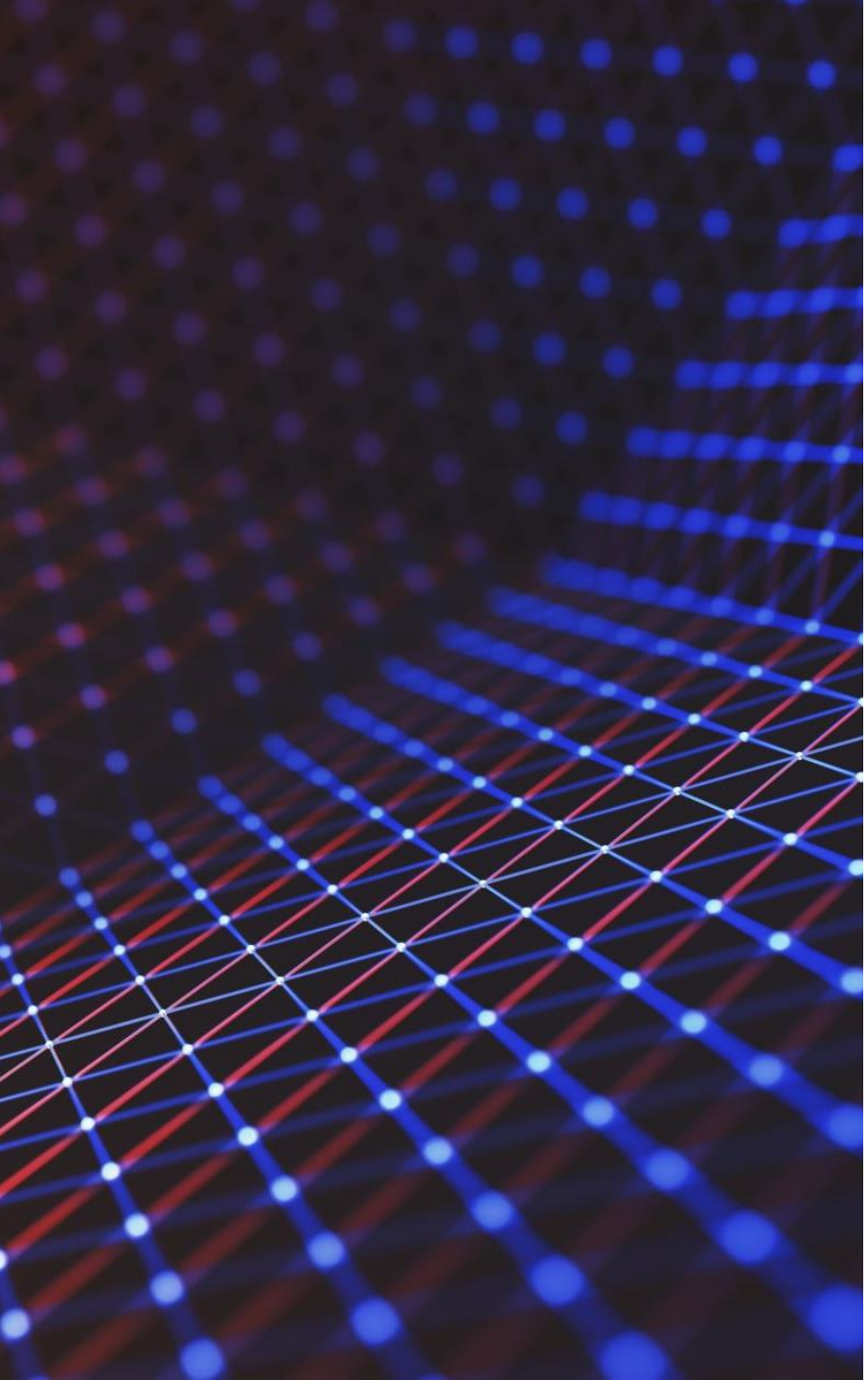


TRANSFORMERS + ESTIMATORS

- ***pipeline*** = chain of ***transformers+estimators***

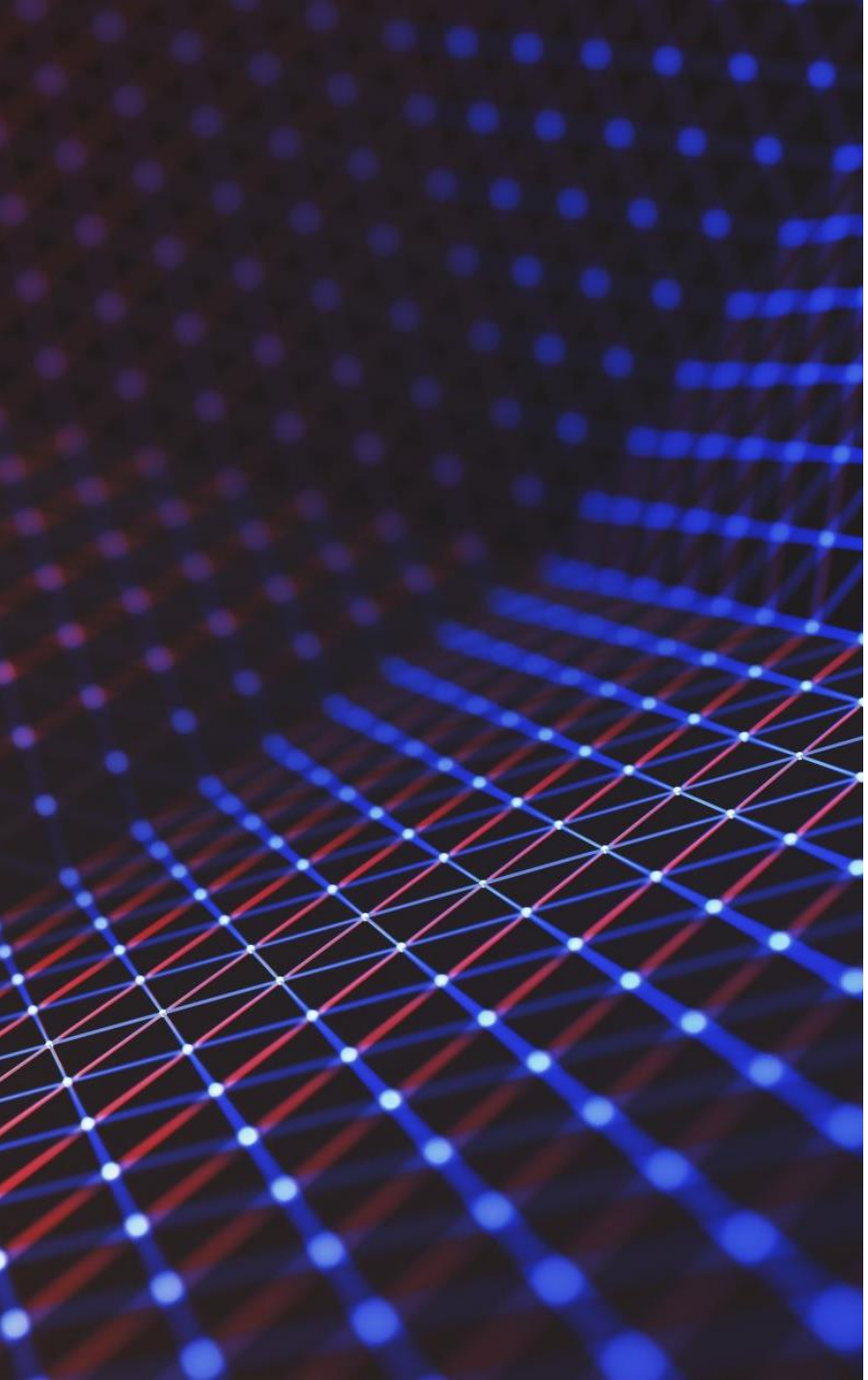
- ***transformer*** : BaseTransformer = alg that can transform one DataFrame into another (for feature transformations), w/ **fit()+transform()**

- ***estimator*** : BaseEstimator = alg (learner) that can be fit on a DataFrame to produce a transformer (ex: classifier/regressor), w/ **fit()**



MODEL FUNCTIONS

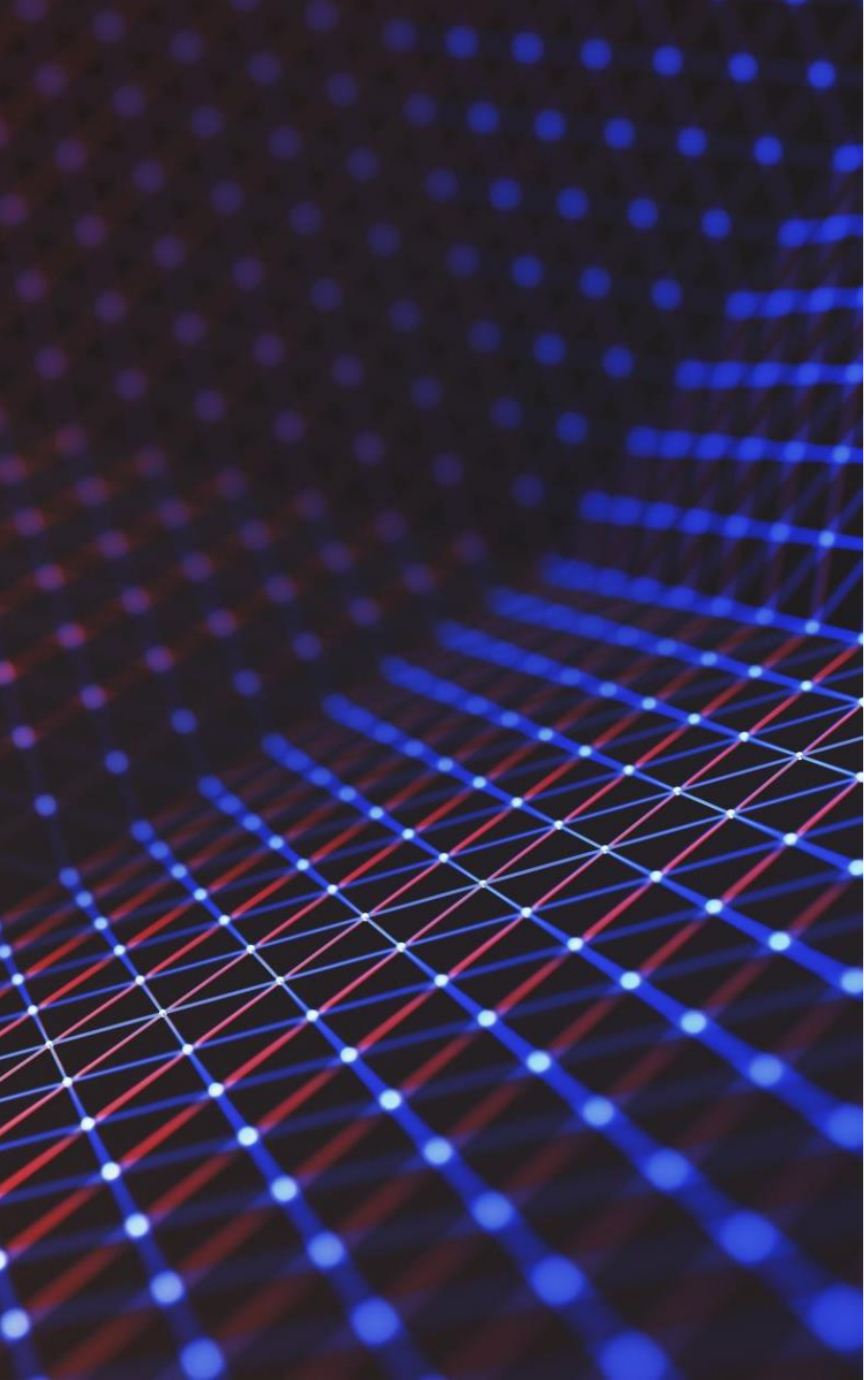
- model.**fit**(X_train, y_train) = learn/estimate/save only params of the transformations internally (like mean, std dev)
- model.**transform**() = applies transformation learned from fit() on data
- model.**score**(X_test, y_test) = evaluate model accuracy using test data, on model fit w/ train data
- y_pred = model.**predict**(X_text) = predict new classification/regression values using test values



REGULARIZATION

- for highly correlated features in linear regression
- penalize the model (adding a ***loss function***) from learning weights that do not generalize well to unseen data → reduces overall model complexity + prevents overfitting

- **LASSO (L1) regression** = min sum(MSE) + weighted L1 norm → line/polynomial
- **Ridge (L2) regression** = min sum(MSE) + weighted L2 norm → line/polynomial
- **ElasticNet regression** = combines the two

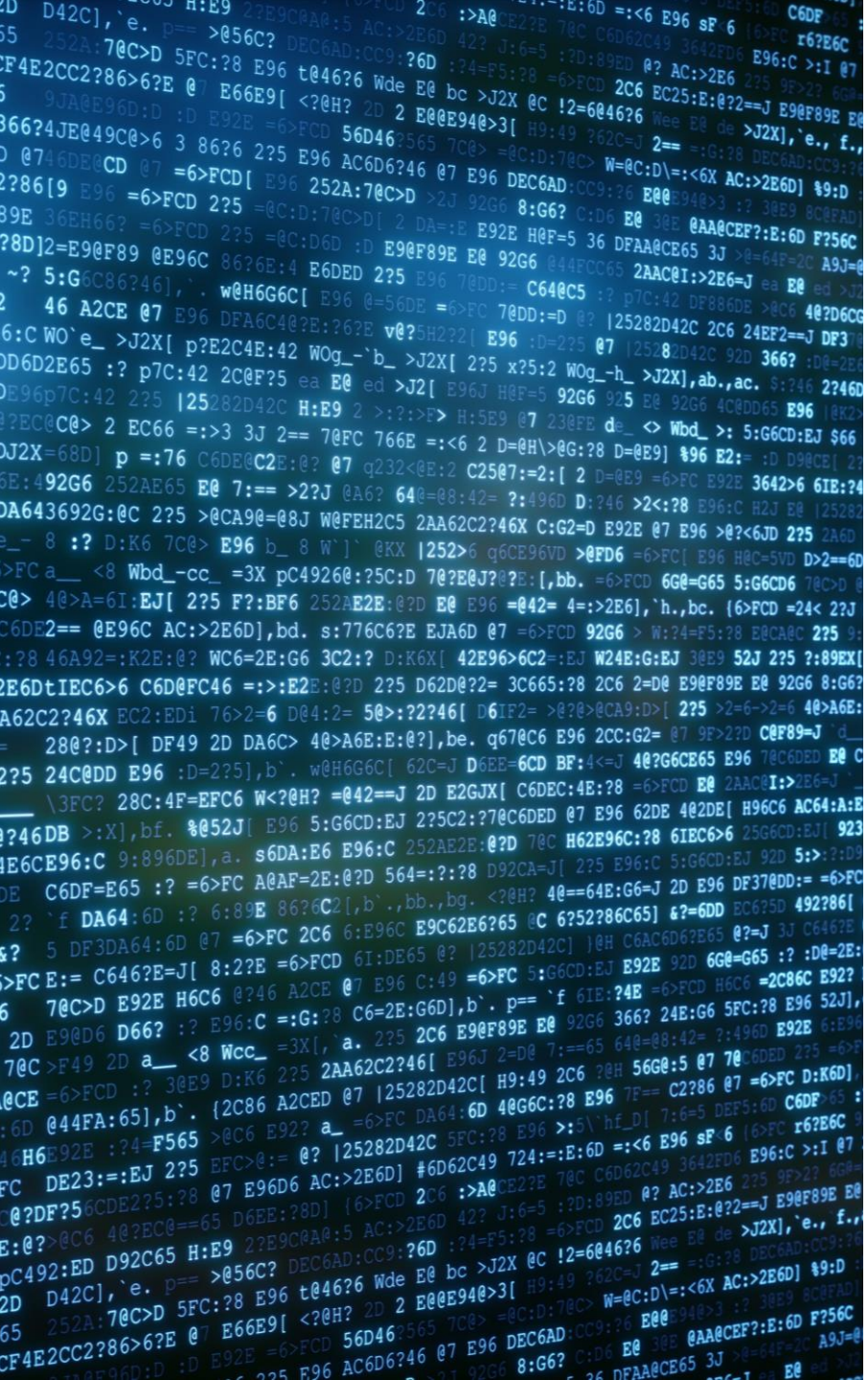


DIMENSIONALITY REDUCTION

- **PCA (Principal Component Analysis)** = unsupervised method that identifies the combination of attributes (principal components) that account for the most variance in the data. It uses linear dimensionality reduction using **SVD (Singular Value Decomposition)**.

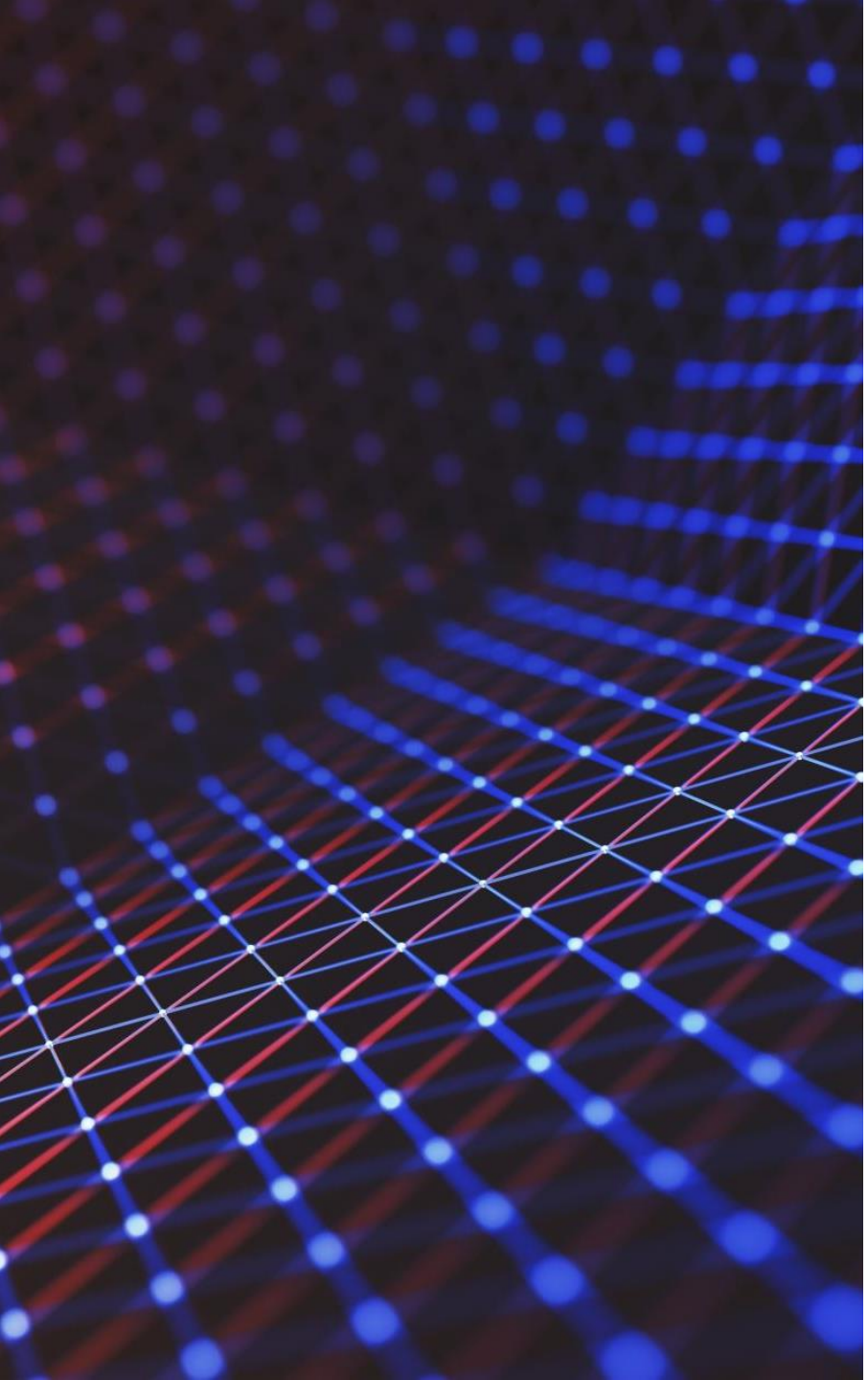
- **LDA (Linear Discriminant Analysis)** = supervised method that identifies attributes that account for the most variance between classes, using Bayes rule. The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix.

- **NCA (Neighborhood Components Analysis)** = supervised method that finds a feature space such that a stochastic nearest neighbor algorithm will give the best accuracy. ML algorithm for metric learning.



CV (CROSS VALIDATION) TECHNIQUES

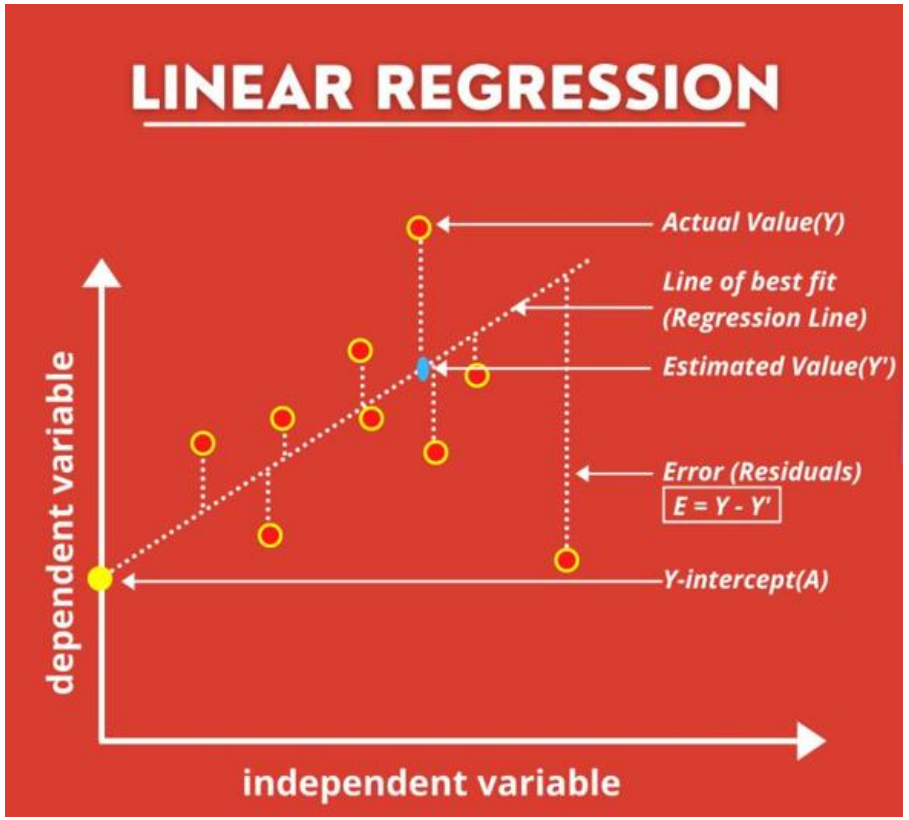
- **Hold-Out** = split ~80/20% between train/test data
- **k-Fold** = for small datasets (time-consumming for k>>)
- **Leave-One-Out (LOOCV)** = k-Fold w/ k = number of samples
- **Leave-p-Out (LpCV)** = ~LOOCV, but sets will overlap if p>1
- **Stratified k-Folds** = k-Fold variation for imbalanced datasets, w/ ~same % of samples of each target in each fold
- **Repeated k-Folds** = split model k times randomly
- **Nested k-Folds** = split also each fold into k, to tune hyperparameters
- **Rolling** = gradually increase TS training w/ another fold
- **Blocked** = also for TSs, jump to another train+test split



HYPERPARAMETER OPTIMIZATION

- **Manual Search**
- **GridSearchCV** = exhaustive grid search
- **RandomizedSearchCV** = random search
- **HalvingGrid/RandomSearchCV** =
- **Bayesian Optimization**
- **Hyperband**

REGRESSION METRICS



- **MAE (Mean Absolute Error)** = normalized, avg diff between predicted and true values
- **RAE (Relative Absolute Error)** = 0..1, abs diffs between predicted and true values
- **MSE (Mean Squared Error)** = cost function to calculate/adjust line
- **RSE (Relative Squared Error)** = 0..1, square of the diffs between predicted and true values
- **RMSE (Root MSE)** = select this, same unit as the label → choose close to 0!
- **R2 score (Coef. of Determination/R-Squared)** = 0..1 → choose closer to 1!

CONFUSION MATRIX FOR BINARY CLASSIFICATION

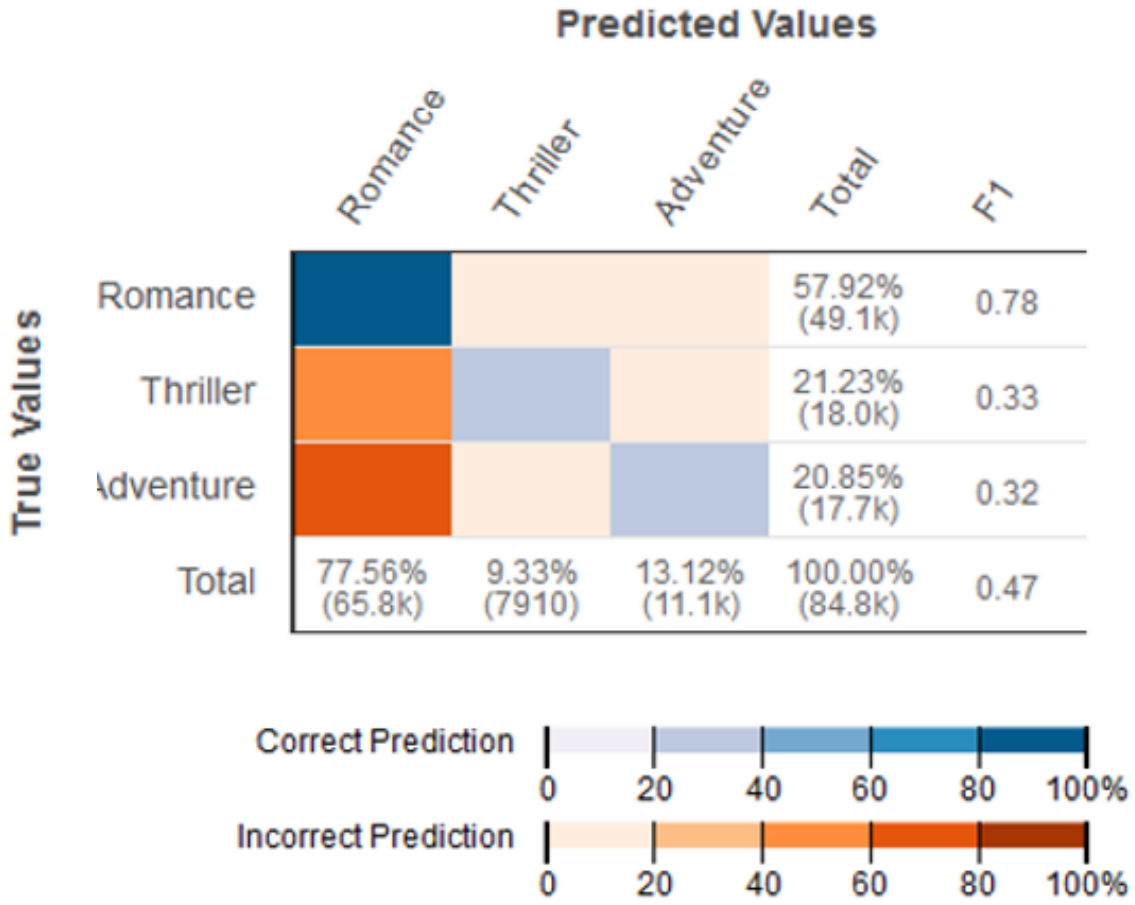
		Predicted: Positive	Predicted: Negative
Actual	Positive	True Positive	False Negative
Actual	Negative	False Positive	True Negative

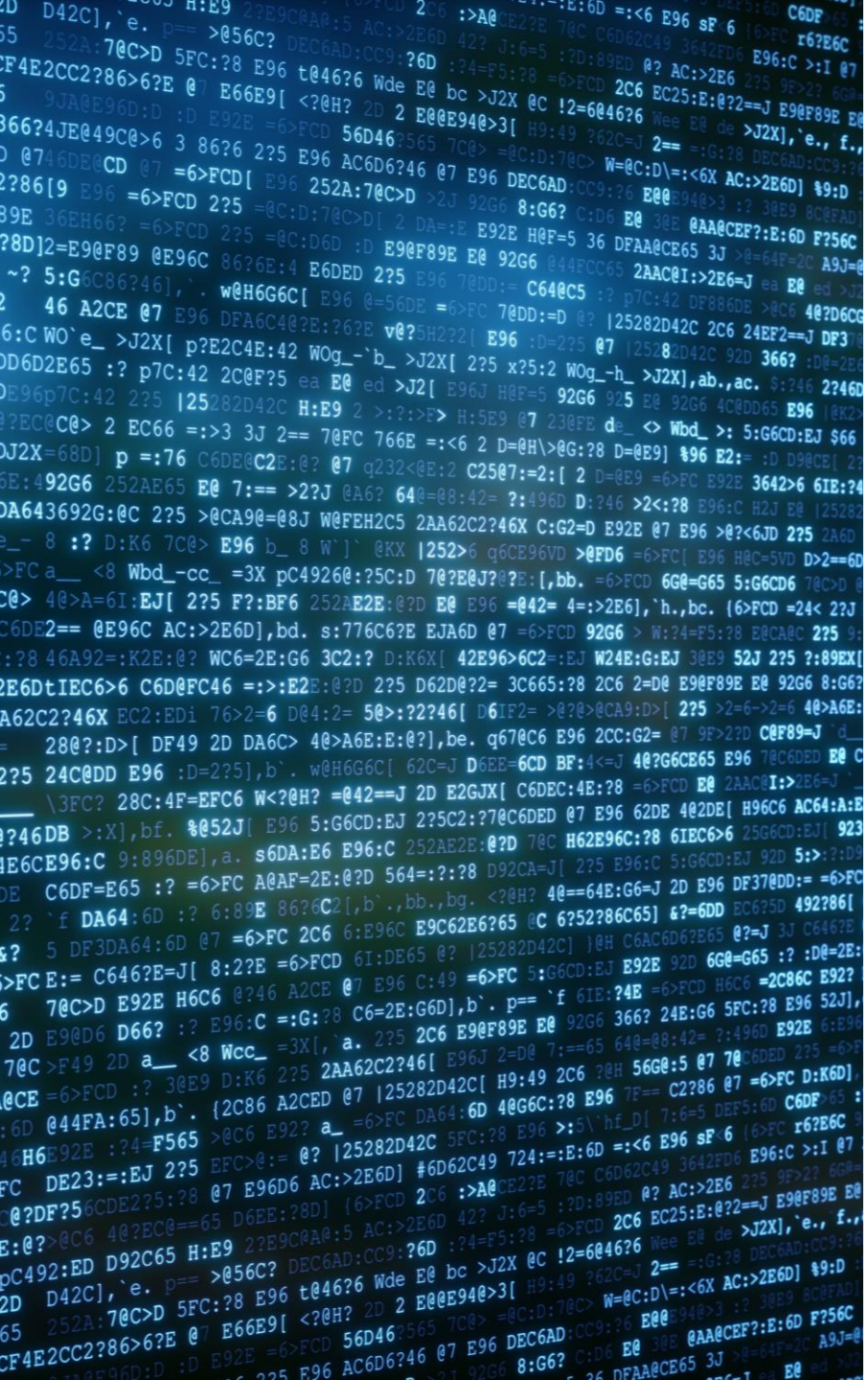
- **Accuracy** = (TP + TN) / (P + N) ← 0..1
- **Precision** = TP / (TP + FP)
- **Recall** = TP / (TP + FN)
- **F1 Score** = TP / (TP + (FP + FN) / 2)

= 2 * (Precision * Recall) / (Precision+ Recall)

- Positives (P) = TP + FN
- Negatives (N) = TN + FP

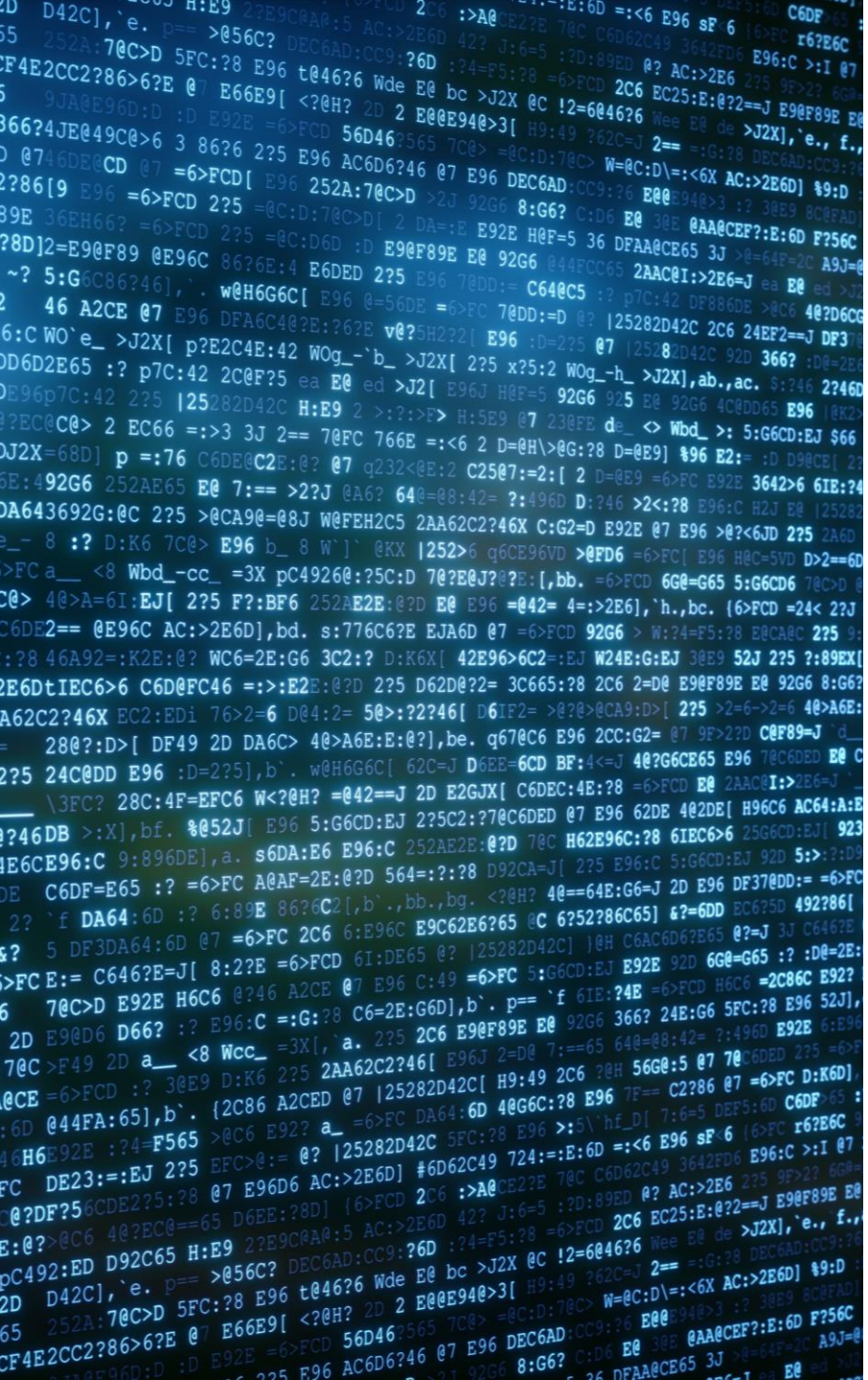
CONFUSION MATRIX FOR MULTICLASS CLASSIFICATION





MODEL INFERENCE

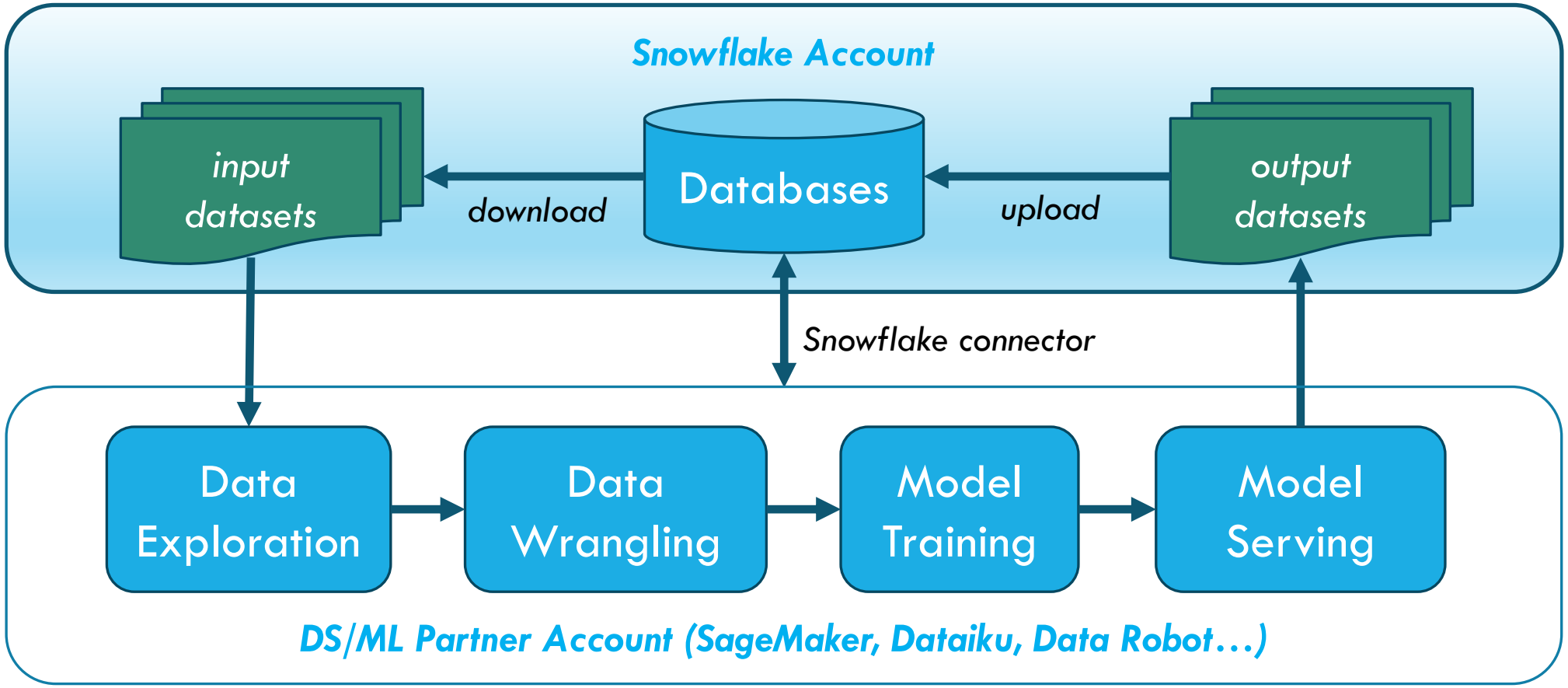
- local file → problem!
- batch/near-real-time
- Pickle
- joblib

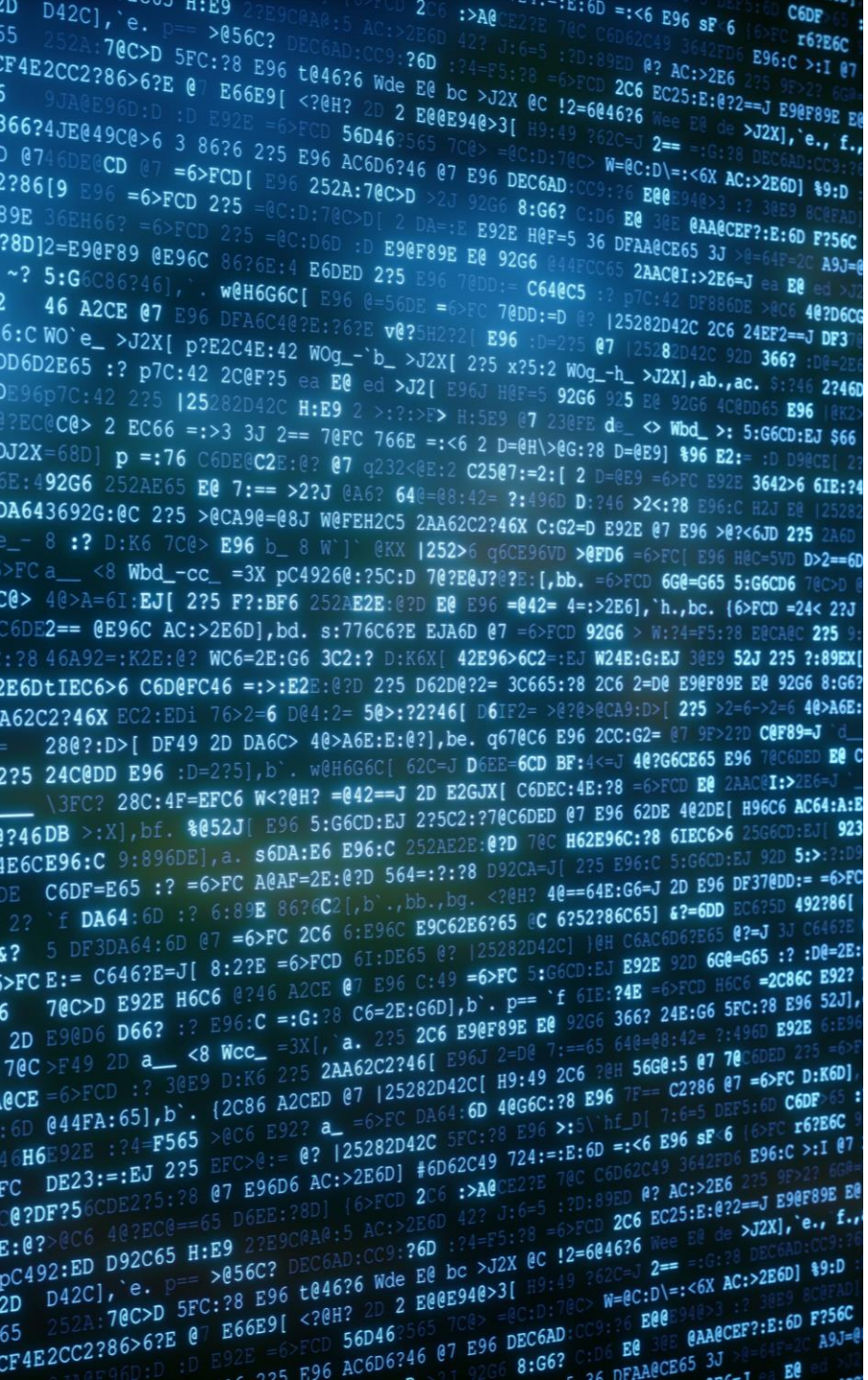


MLOPS

- **model deployment** = package+deploy w/ CI/CD pipelines ← GitHub/Bitbucket, Docker/ECR
  - ***as a service (live-scoring)***
  - ***embedded*** = packaged into a published app
- **model testing** = on deployment
  - ***shadow testing***
  - ***A/B testing***
- **model inference** = operate w/ predictions, see app integration ← *model serving*
- **model monitoring** = w/ *perf monitoring+logging*
- **model iteration** = update w/ new data ← *retrain+validate model*

SNOWFLAKE INTEGRATIONS WITH DS/ML PARTNERS





DATA SCIENCE & MACHINE LEARNING SNOWFLAKE PARTNERS

- **Dataiku** → Accelerating Data Science with Snowflake and Dataiku quickstart
- **DataRobot** → Accelerating Machine Learning with Snowflake and DataRobot quickstart
- **Amazon SageMaker** → Deploying Models from AzureML and Sagemaker to Snowpark ML quickstart
- **H2O.ai** → AutoML with Snowflake and H2O Driverless AI quickstart
- **Hex** → Building and deploying a Time-Series forecast with Hex and Snowflake quickstart
- **Zepl** → Time-series Forecasting with Zepl quickstart
- **Databricks, Apache Spark, Alteryx,** Big Squid, BoostKPI, Domino, Qubole, SAS, Tellius…

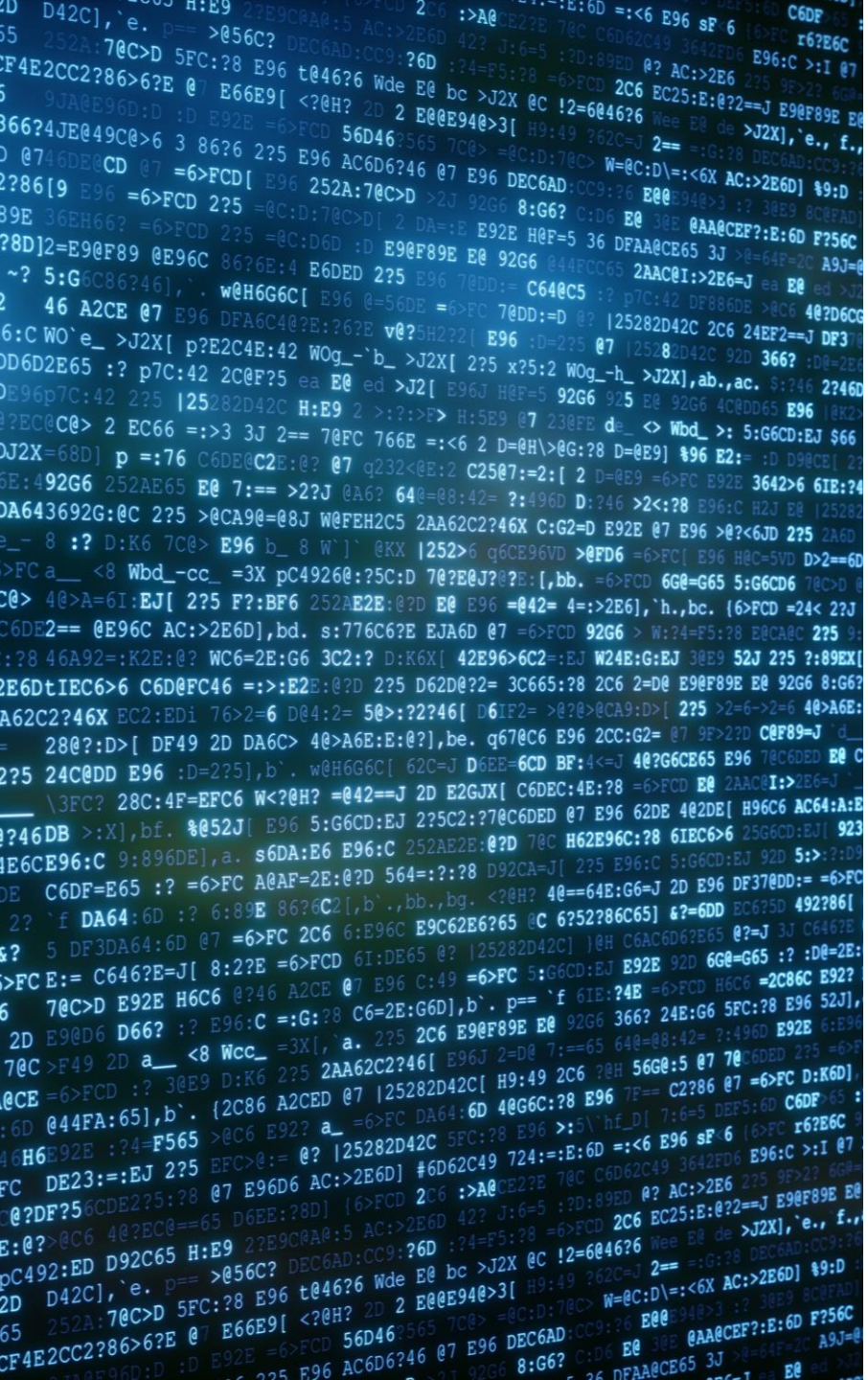
SNOWPARK

DataFrame API

Functions & Procedures

Machine Learning

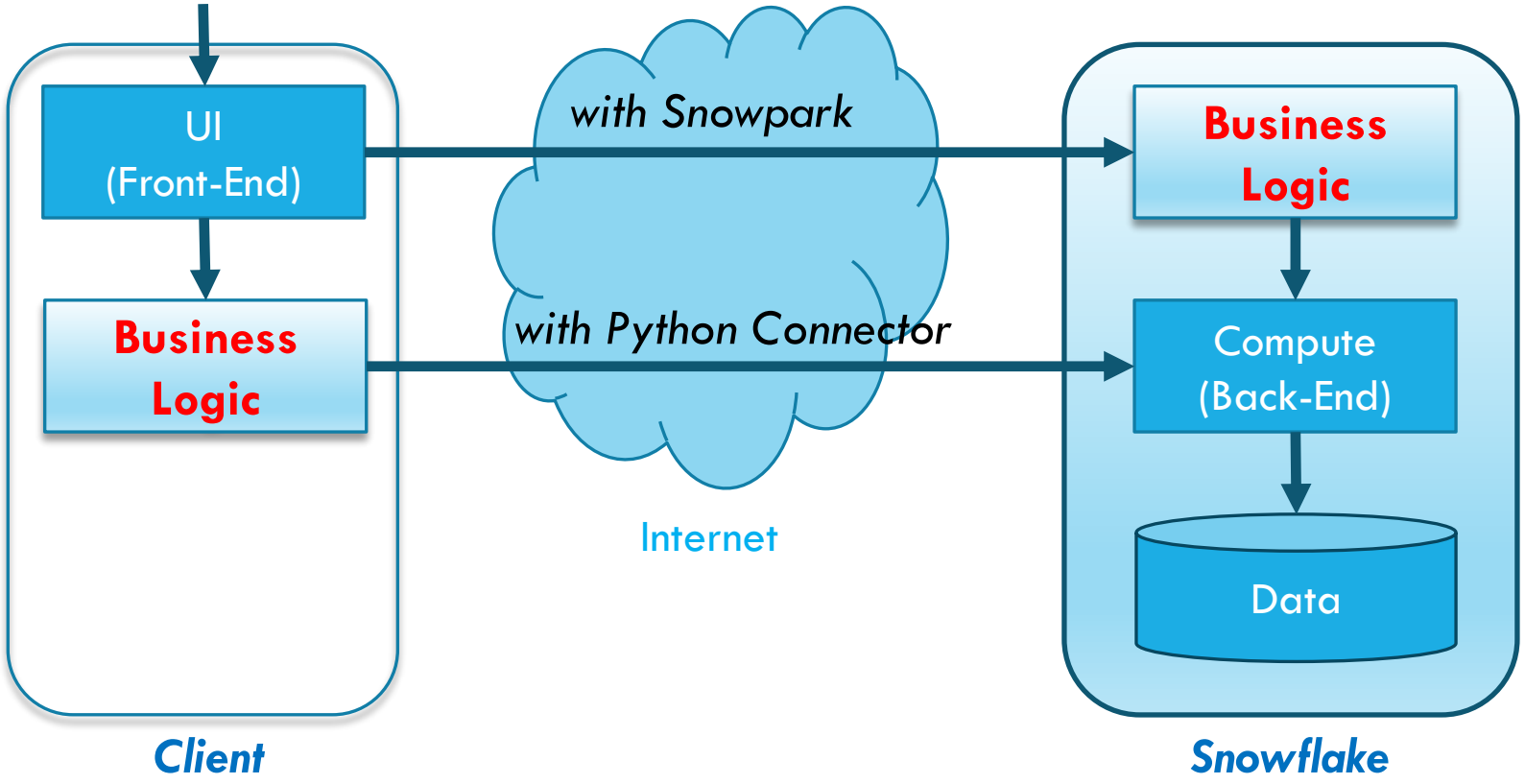
Container Services



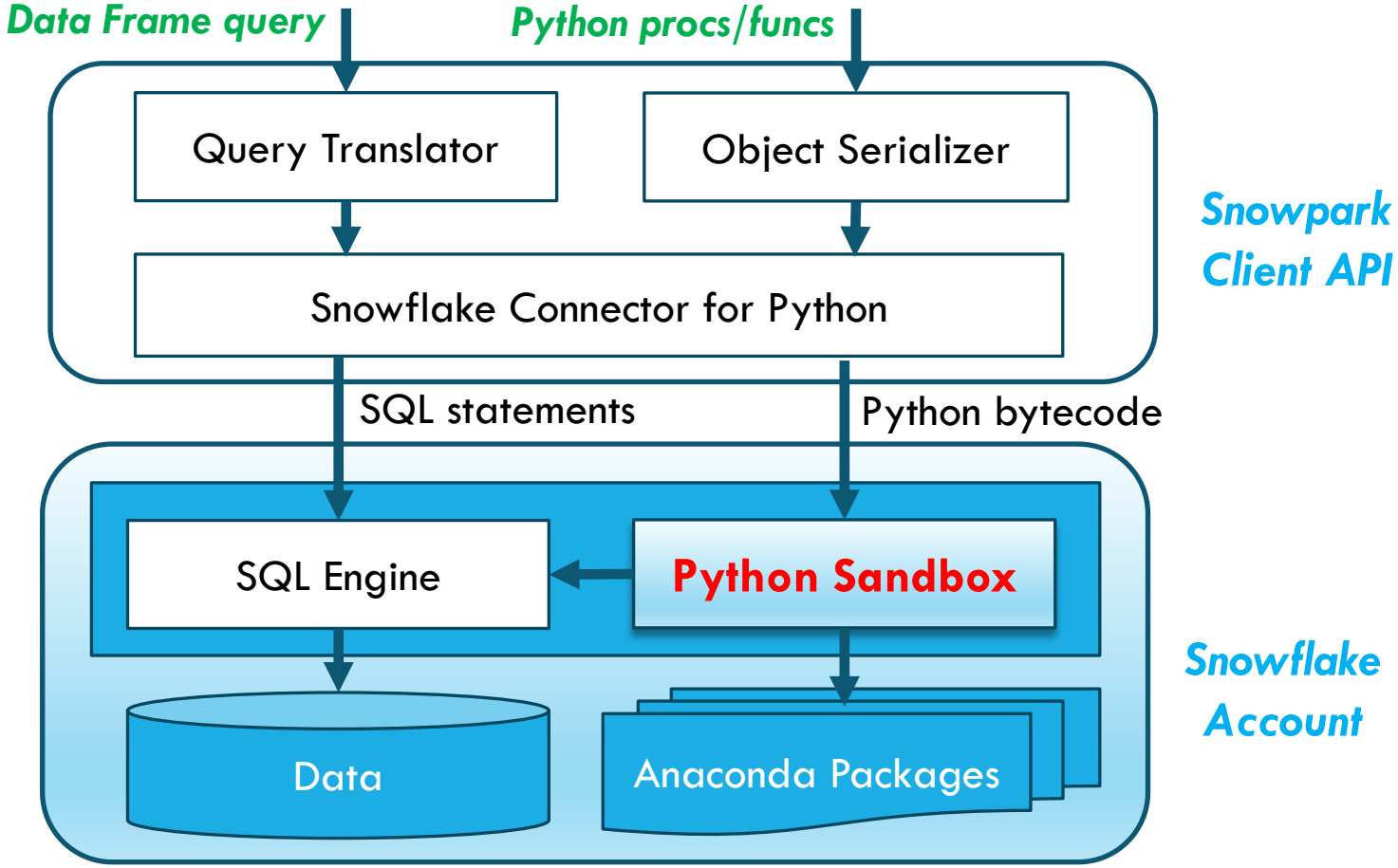
SNOWPARK

- ***DataFrame API*** = w/ DataFrame class as better alt for Snowflake to Pandas, inspired by PySpark data frames
- ***Functions & Procedures*** = @udf/udtf/sproc Python decorators + vectorized UDFs
- ***Machine Learning*** (Snowpark ML) = w/ Modeling + Operations + Data Access
- ***Container Services*** (SPCS) = in PrPr, w/ Docker containers & orchestration, including GPUs for training models and LLMs.

SNOWPARK VS PYTHON CONNECTOR



SNOWPARK FOR PYTHON ARCHITECTURE



FUNCTIONS & PROCEDURES THROUGH SNOWPARK

- ***creating***

  sproc/udf/udtf(lambda: ..., [**name**="..."], ...) ← ***anonymous/named***

  sproc/udf/udtf.**register**(**name**="...", **is_permanent**=True, ...) ← ***registered***

  @sproc/@udf/@udtf(**name**="...", **is_permanent**=True, ...) ← ***registered***

- ***calling***

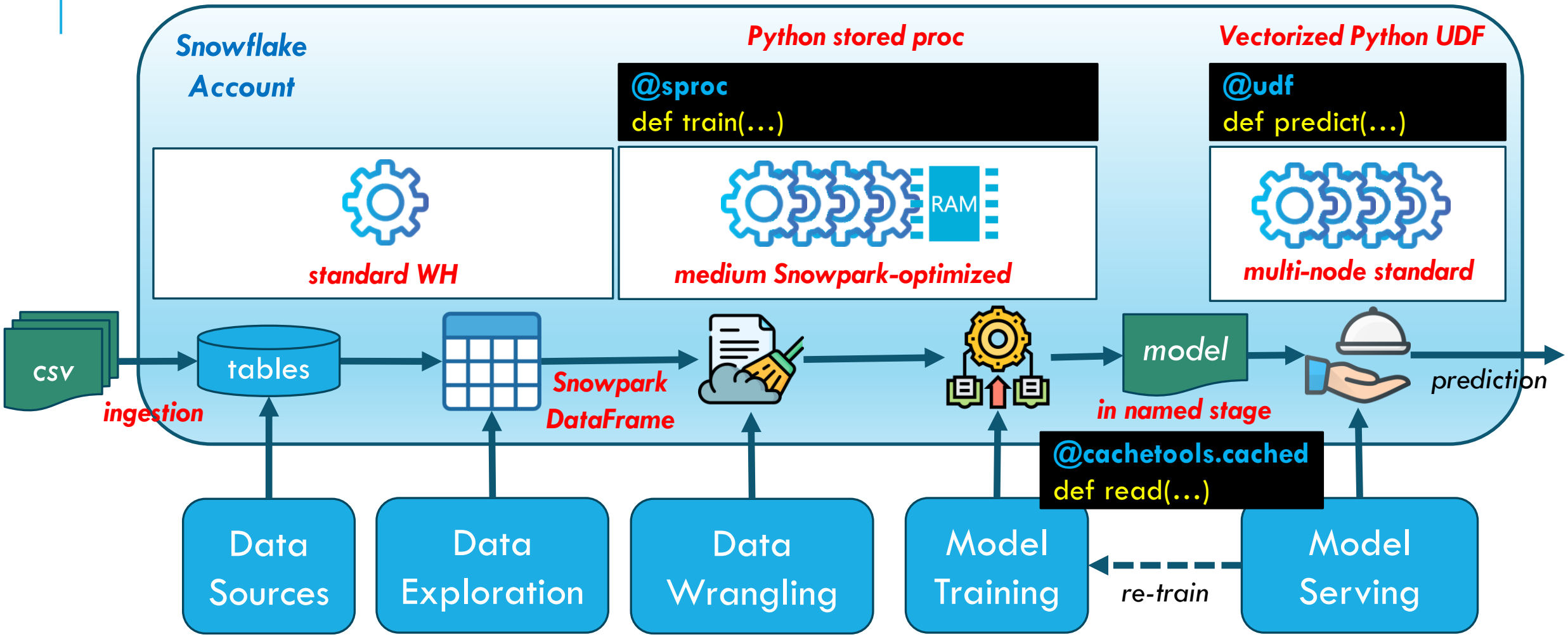
  name(...) / fct = **function**("name") ← by name/function pointer

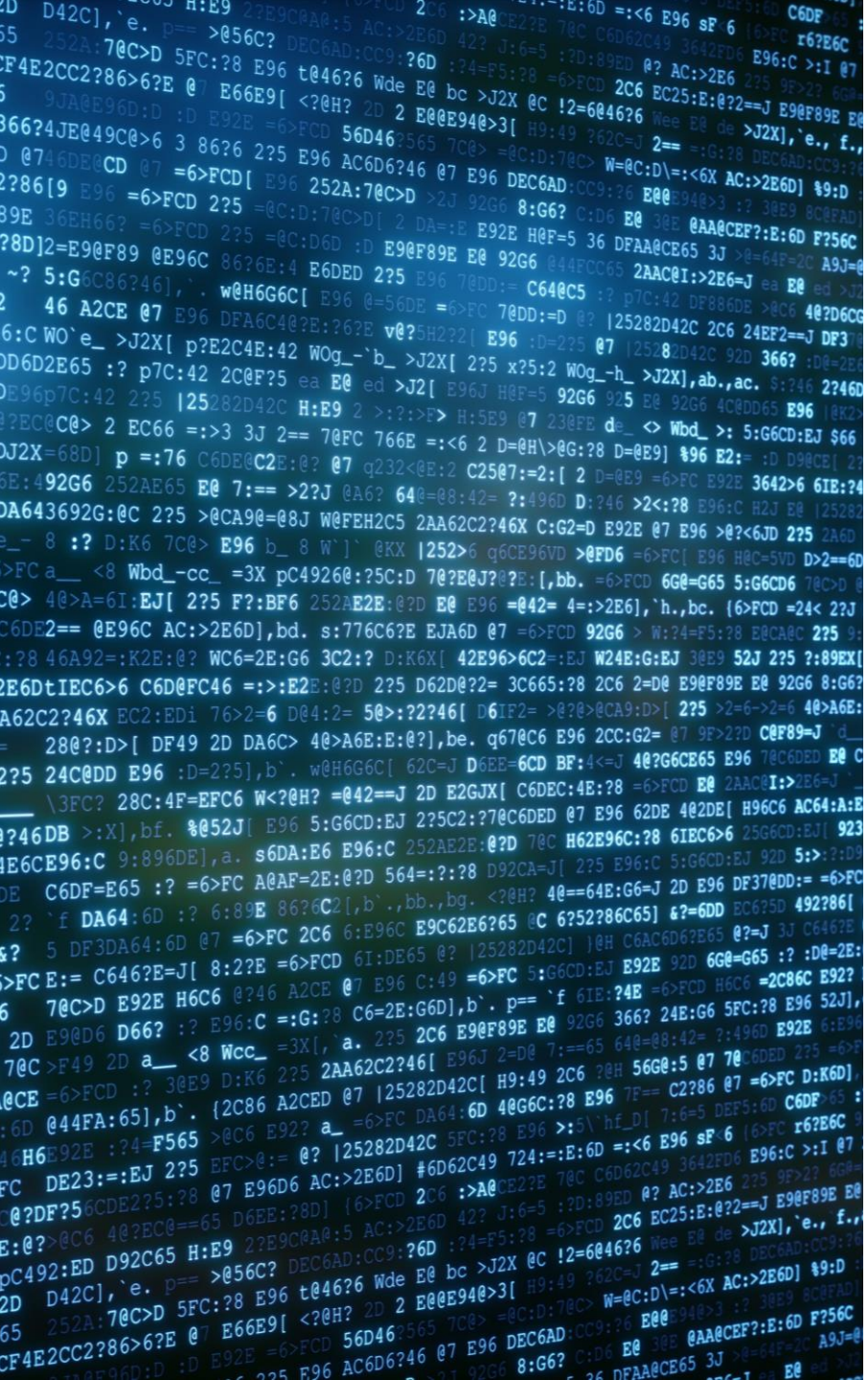
  session.**call/call_function/call_udf**("name", ...) ← SP/UDF

  session.**table_function**(...) / *dataframe*.**join_table_function**(...) ← UDTF

  session.**sql**("**call** name(...)").**collect**() ← SP

ML PIPELINES USING SNOWPARK (NO CORTEX)





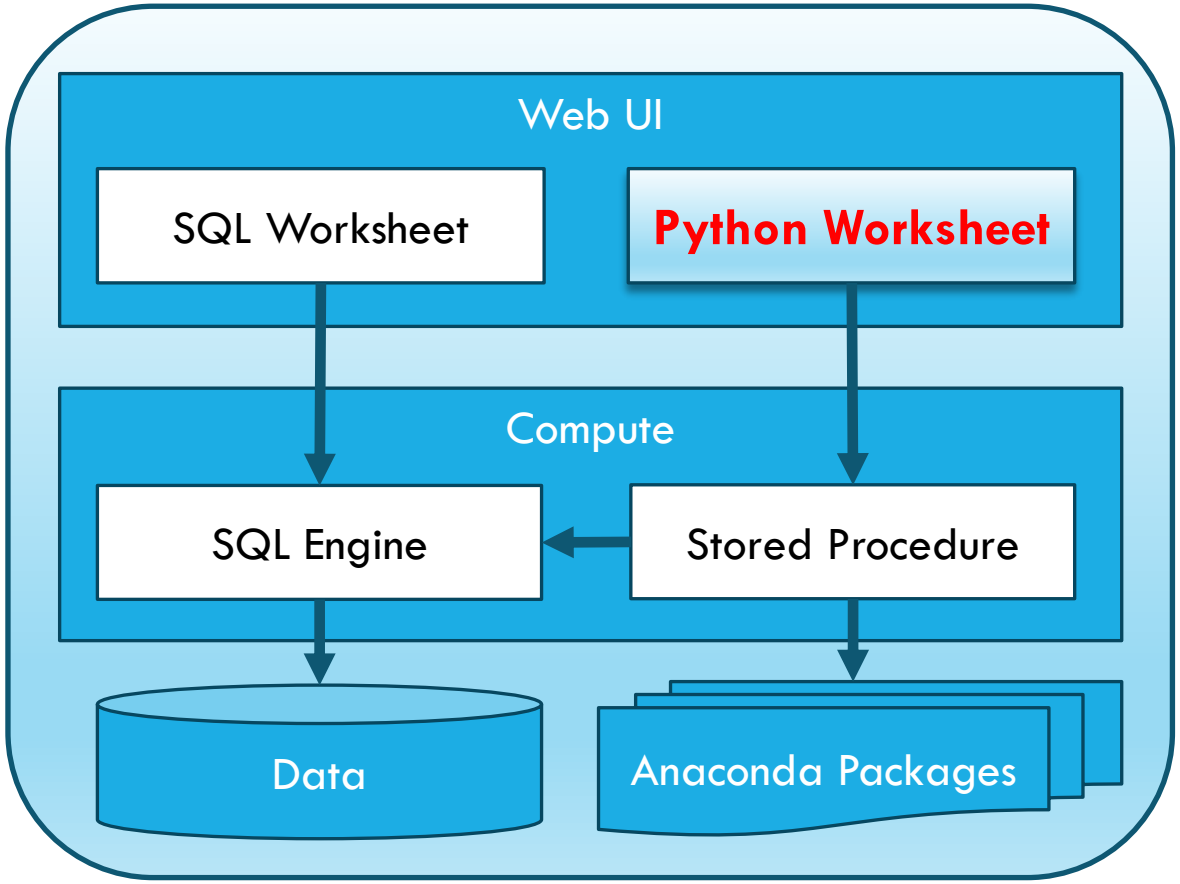
SYNTHETIC DATA GENERATION

- GENERATOR → rows/time_limit
- RANDOM functions → cells
- Faker library (for fake but realistic data)
- through custom code (Python)

SYNTHETIC DATA GENERATION

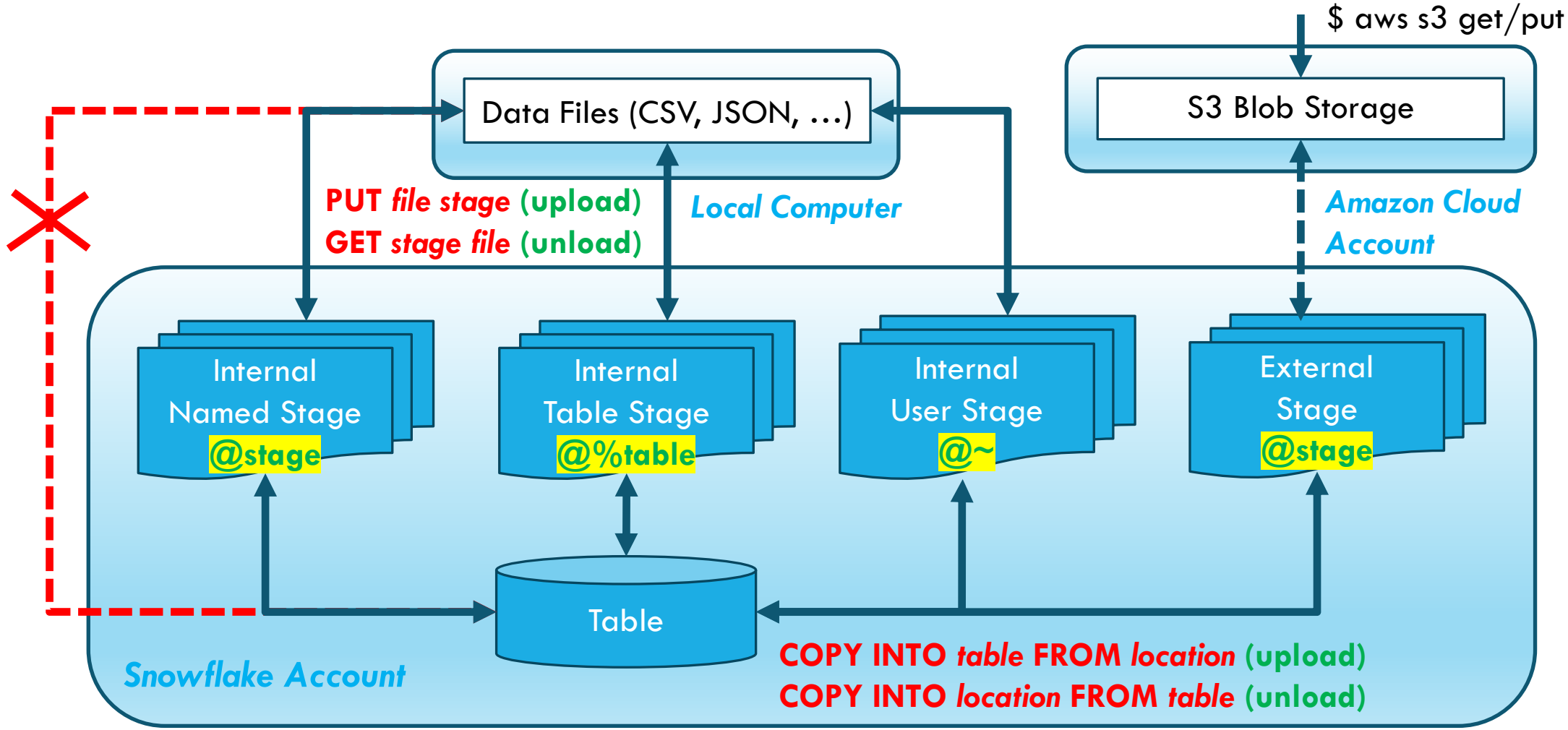
- **SELECT ... FROM TABLE(GENERATOR([rowcount], [timelimit]))** ← rows (w/o columns)
- ***Random*** ← deterministic values
  - RANDOM/RANDOMSTR ← 64-bit integer/string with length
  - UUID_STRING ← UUID
- ***Controlled Distribution*** ← for unique ID values
  - NORMAL/UNIFORM/ZIPF ← number w/ specific distribution/integer
  - SEQ1/SEQ2/SEQ4/SEQ8 ← sequence of integers

PYTHON WORKSHEETS

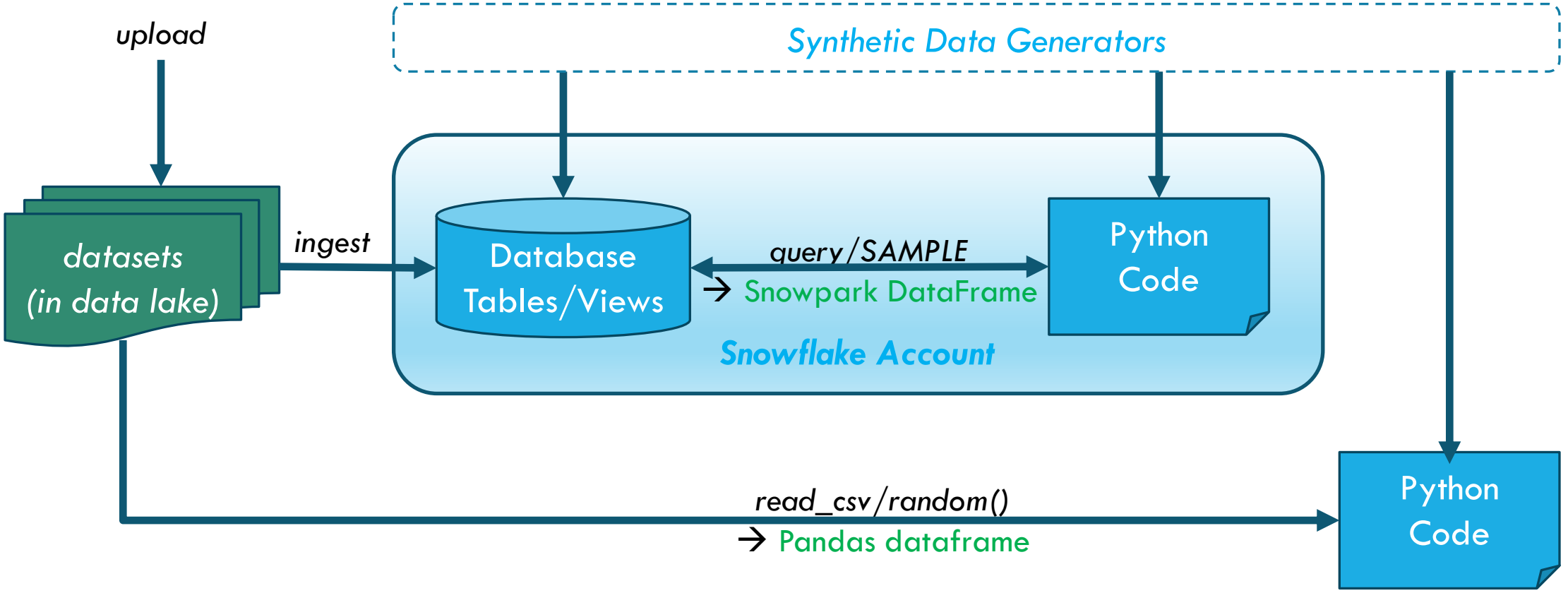


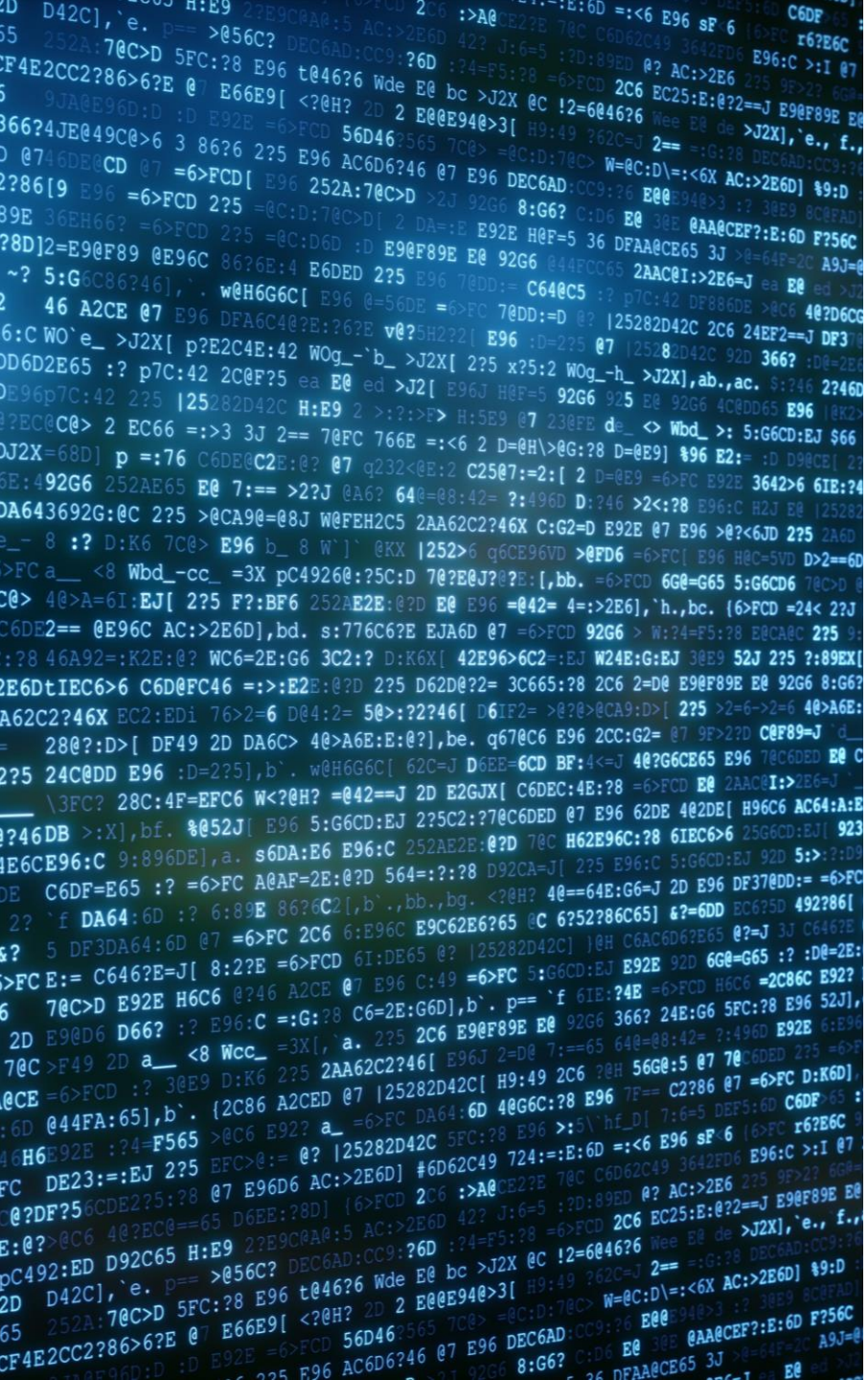
Snowflake Account

UPLOADING/UNLOADING DATA IN SNOWFLAKE



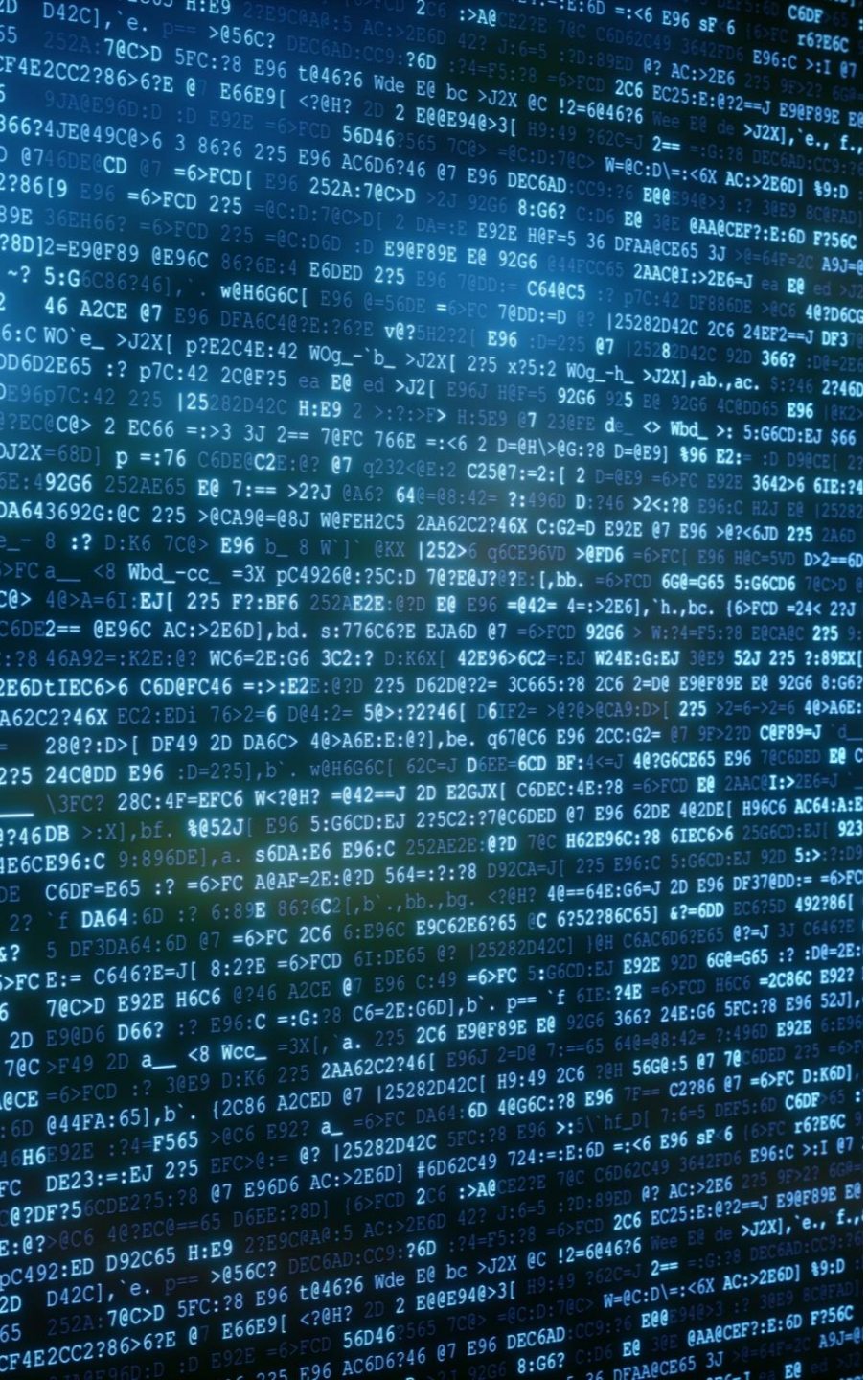
DATA COLLECTION (FROM DATA LAKEHOUSE)





SAMPLE DATA EXTRACTION

- ***sample datasets***
  - SNOWFLAKE_SAMPLE_DATA → builtin
  - Snowflake Marketplace → free/public
- ***sample data extraction***
  - SAMPLE/TABLESAMPLE → rows/random



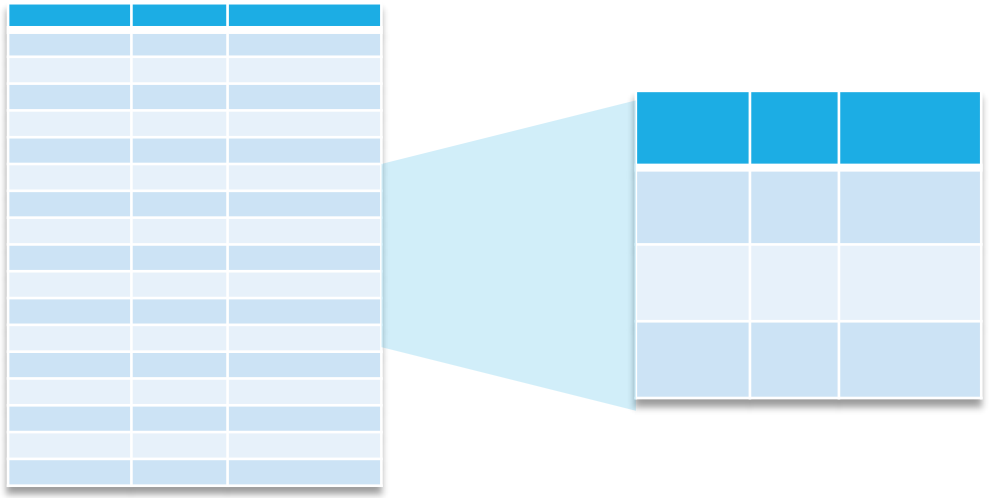
SNOWFLAKE_SAMPLE_DATA

- TPCH_SF1 = ~M elements
- TPCH_SF10 = ~10 x M elements
- TPCH_SF100 = ~100 x M elements
- TPCH_SF1000 = ~B elements

- TPCDS_SF10TCL = 7 fact tables+17 dims, 10 TB
- TPCDS_SF100TCL = 100 TB

SAMPLE DATA EXTRACTION

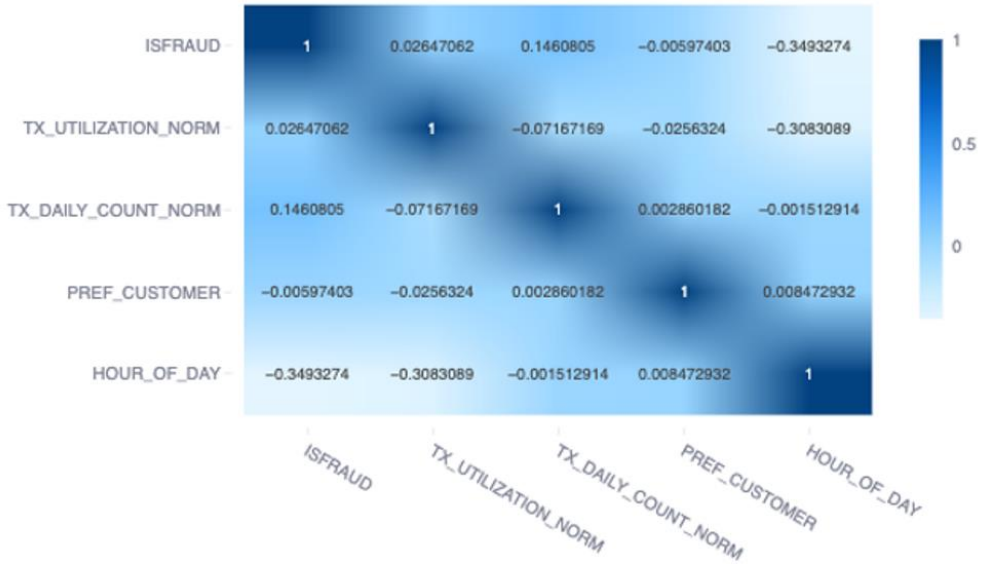
- **SELECT ... FROM ... SAMPLE** ← extracts subset of rows from a table
  - BERNOULLI (~ROW) | SYSTEM (~BLOCK) ← sampling method
  - <probability> | <n> ROWS ← sample size
  - REPEATABLE (~SEED) (<seed>) ← seed value (to make it deterministic)



SNOWFLAKE NOTEBOOKS

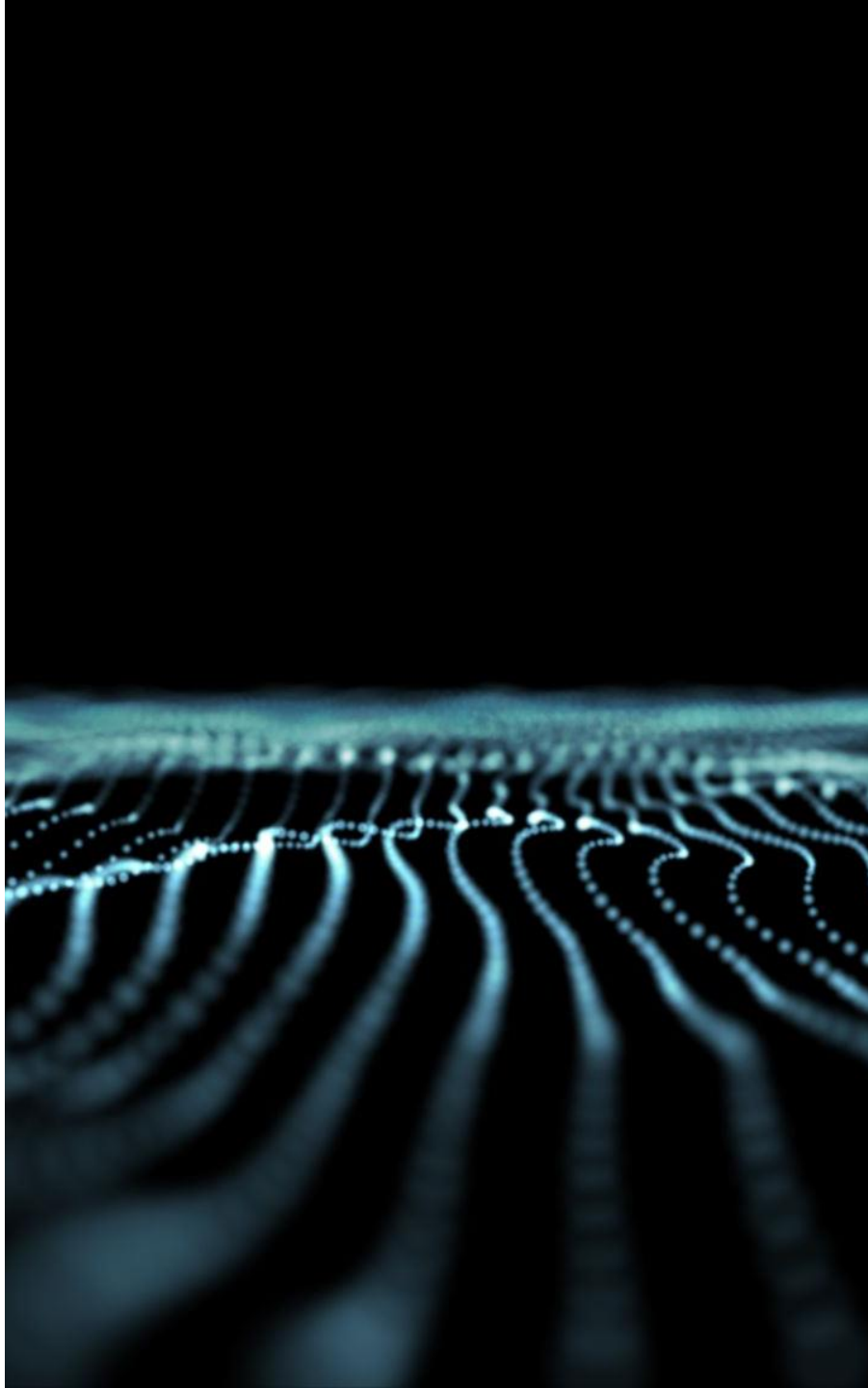
Python ▾ as plot_correlation

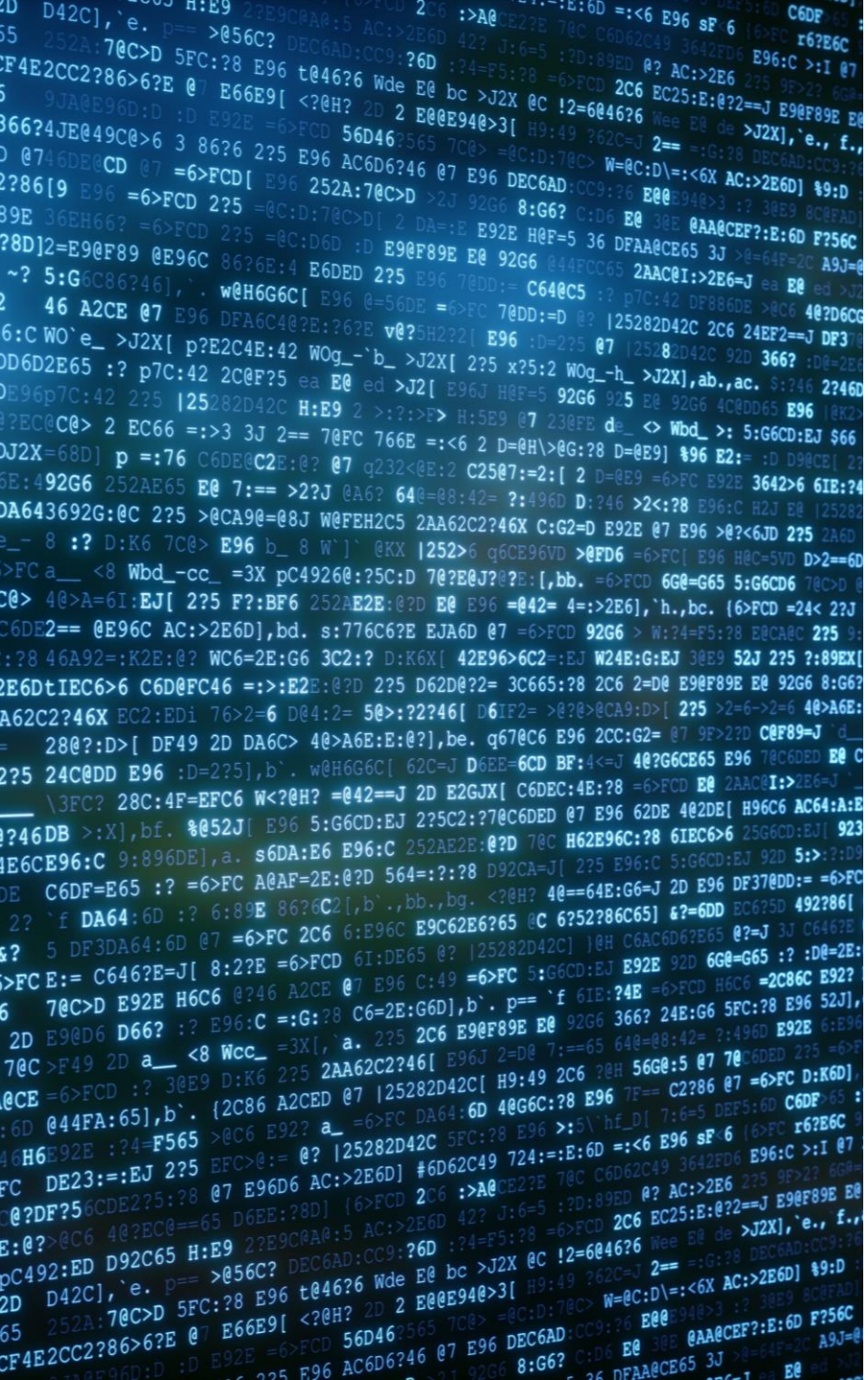
```
1  import plotly.express as px
2
3  fig = px.imshow(corr_df, text_auto=True, aspect="auto")
4
5  # you can choose "streamlit" theme if preferred
6  st.plotly_chart(fig, theme="streamlit")
```



SNOWFLAKE NOTEBOOKS

- in PrPr
- write/execute code in SQL/Python at the cell level directly in Snowsight
- process/experiment with data in Snowpark
- train/deploy models w/ Snowpark ML
- ***visualize results*** with Streamlit chart elements (unlike **Colab, Studio Lab, Kaggle**)
- ***run SQL queries*** w/o any SQL extension/magic command in Jupiter (like %sql for ipython-sql)
- no need to pay for notebooks from third-party partners like **Deepnote, Hex, Zepl, Count**

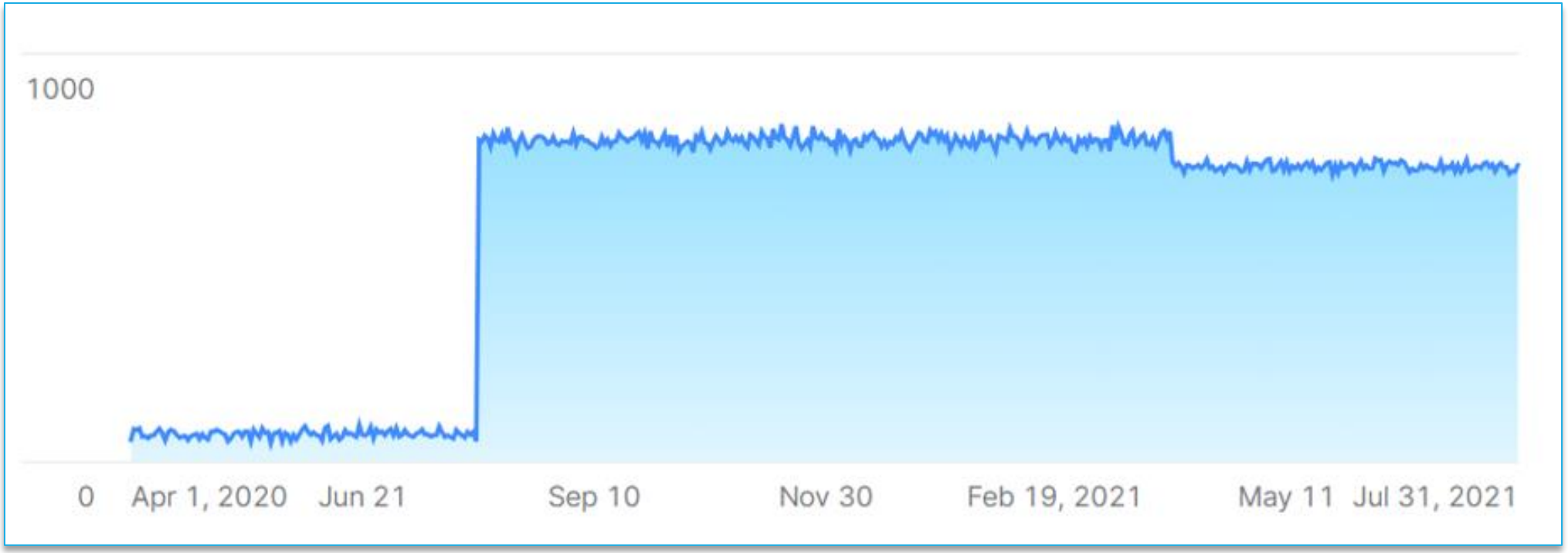




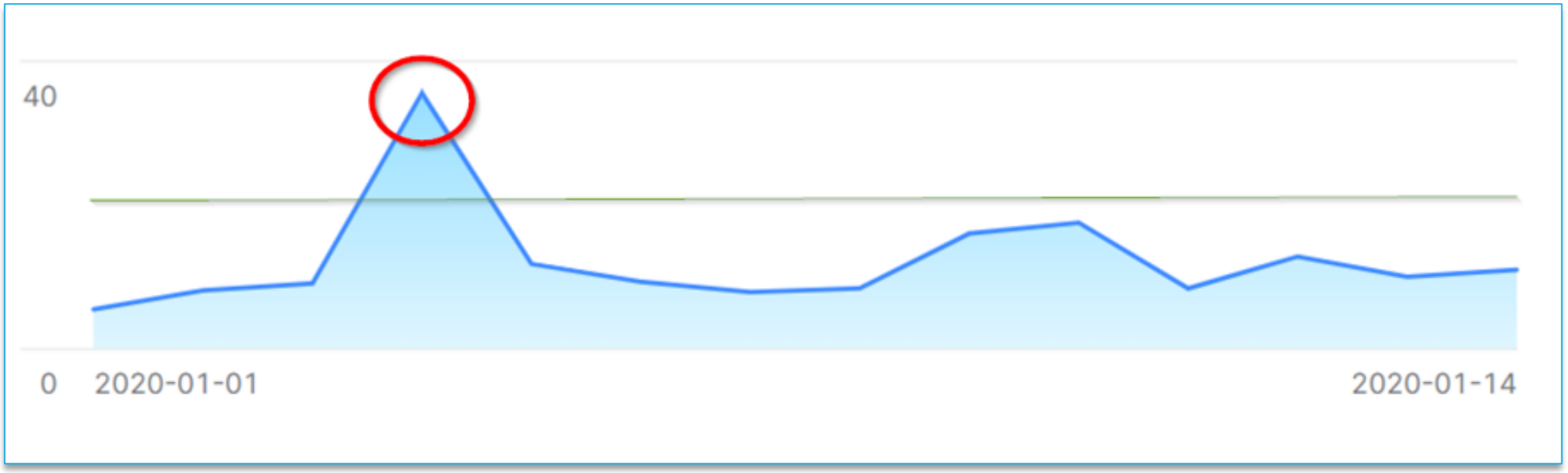
EXPLORATORY DATA ANALYSIS IN SNOWFLAKE

- **Snowsight Charts and Dashboards**
- **Snowsight Data Profiling**
- **Streamlit in Snowflake** (and Native Apps, local Streamlit web apps)
- **Snowflake Notebooks** (incoming)
- *free third-party alts*: **Google Colab, Amazon SageMaker Studio Lab, Kaggle...**
- *paid third-party alts*: **Deepnote, Hex, Zepl, Count...**

TIME SERIES CHART IN SNOWSIGHT (FOR PREDICTIONS)

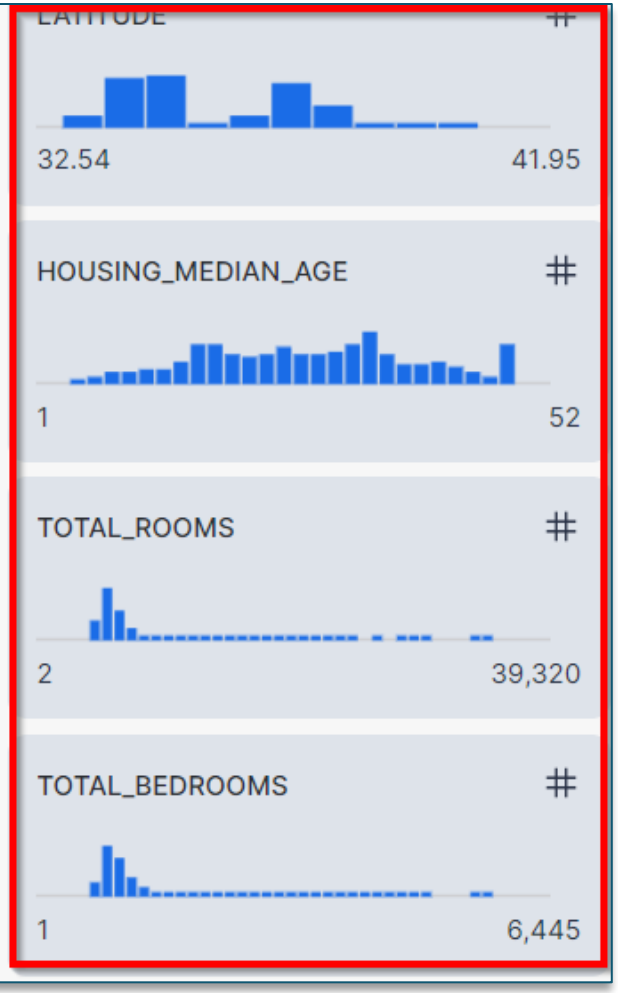


LINE CHART IN SNOWSIGHT (FOR ANOMALY DETECTION)



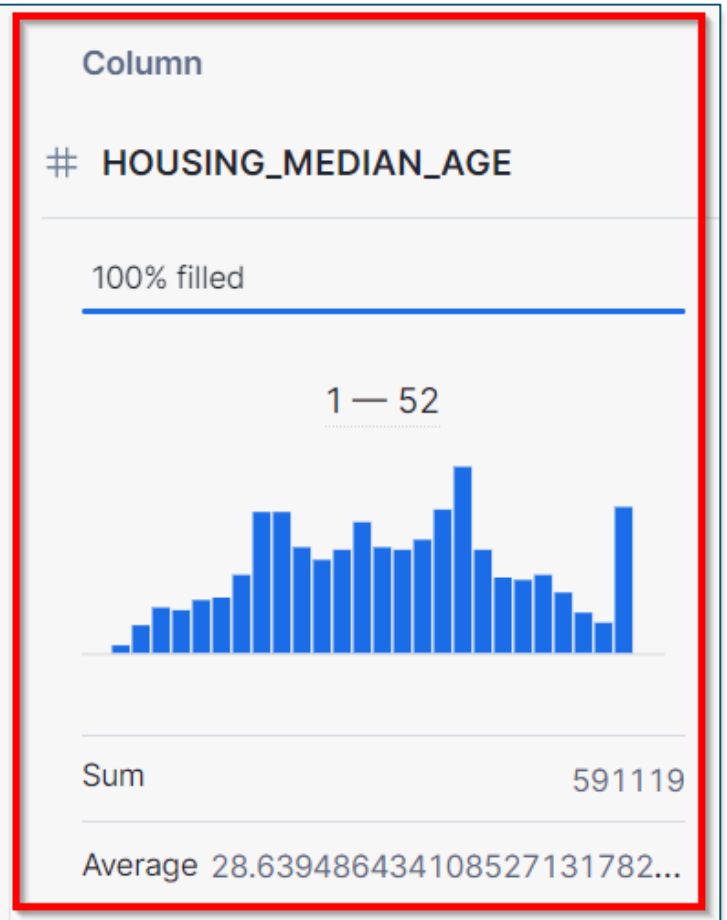
DATA PROFILING: WHOLE QUERY RESULT

	LONGITUDE	LATITUDE	HOUSING_MEDIAN_AGE
1	-122.23	37.88	41.0
2	-122.22	37.86	21.0
3	-122.24	37.85	52.0
4	-122.25	37.85	52.0
5	-122.25	37.85	52.0
6	-122.25	37.85	52.0
7	-122.25	37.84	52.0
8	-122.25	37.84	52.0
9	-122.26	37.84	42.0
10	-122.25	37.84	52.0
11	-122.26	37.85	52.0
12	-122.26	37.85	52.0



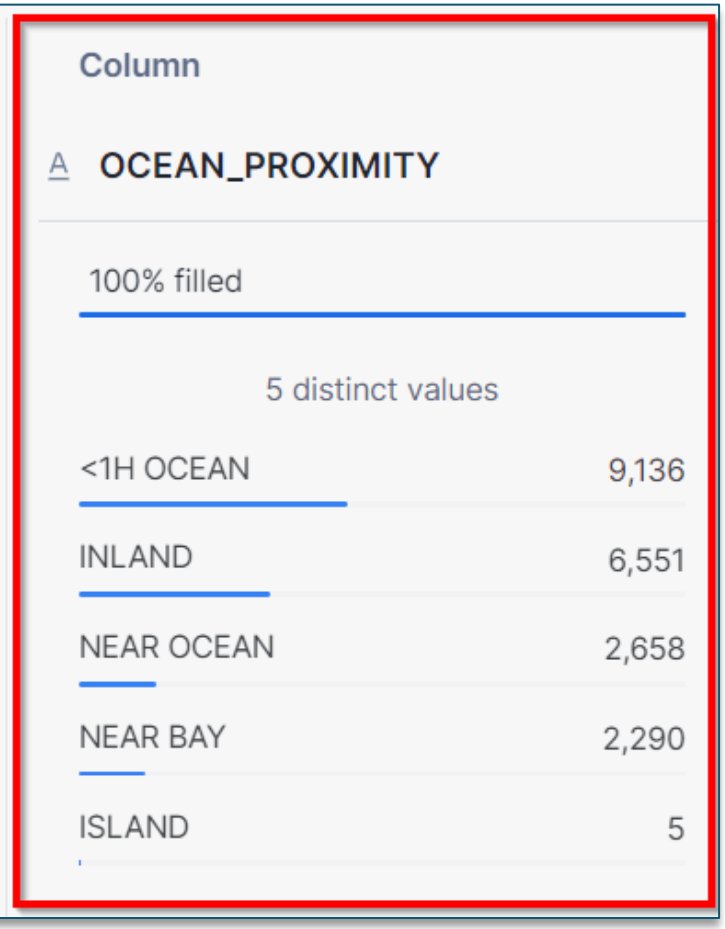
DATA PROFILING: NUMERIC COLUMN

	LONGITUDE	LATITUDE	HOUSING_MEDIAN_AGE
1	-122.23	37.88	41.0
2	-122.22	37.86	21.0
3	-122.24	37.85	52.0
4	-122.25	37.85	52.0
5	-122.25	37.85	52.0
6	-122.25	37.85	52.0
7	-122.25	37.84	52.0
8	-122.25	37.84	52.0
9	-122.26	37.84	42.0
10	-122.25	37.84	52.0
11	-122.26	37.85	52.0



DATA PROFILING: CATEGORICAL COLUMN

	I_INCOME	MEDIAN_HOUSE_VALUE	OCEAN_PROXIMITY ···
1	8.3252	452600.0	NEAR BAY
2	8.3014	358500.0	NEAR BAY
3	7.2574	352100.0	NEAR BAY
4	5.6431	341300.0	NEAR BAY
5	3.8462	342200.0	NEAR BAY
6	4.0368	269700.0	NEAR BAY
7	3.6591	299200.0	NEAR BAY
8	3.1200	241400.0	NEAR BAY
9	2.0804	226700.0	NEAR BAY
10	3.6912	261100.0	NEAR BAY
11	3.2031	281500.0	NEAR BAY



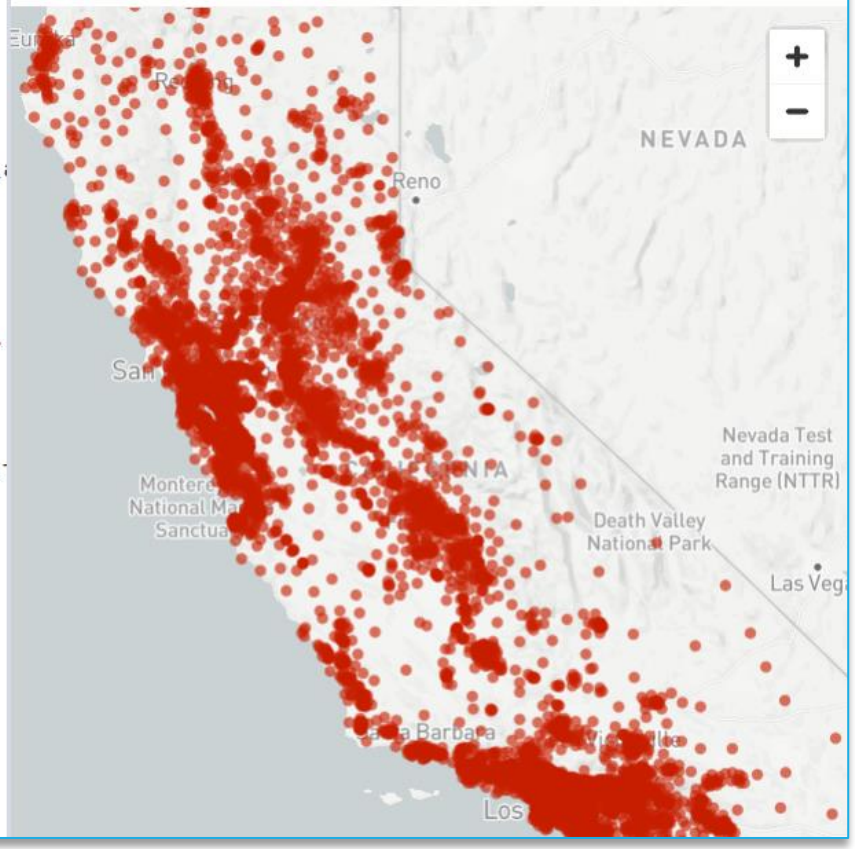
MAP CHART
WITH STREAMLIT IN SNOWFLAKE

Streamlit Apps housing

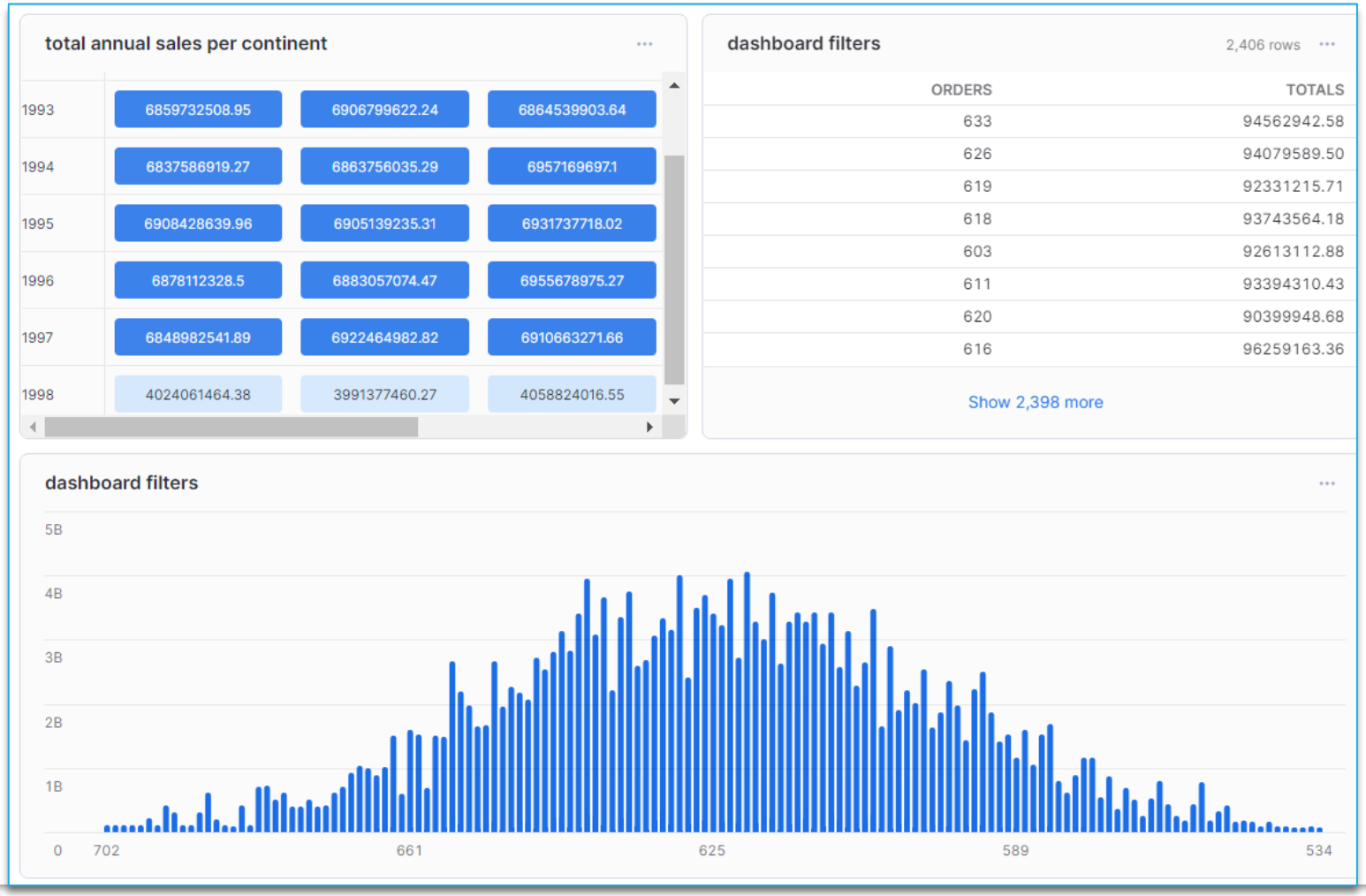
Share Run

Packages

```
1  import streamlit as st
2  from snowflake.snowpark.context import get_
3
4  session = get_active_session()
5  query = """SELECT latitude::float as lat,
6     longitude::float as lon,
7     round(median_house_value / 10000) as val
8  FROM TEST.PUBLIC.HOUSING
9  ORDER BY val DESC"""
10 df = session.sql(query).collect()
11 st.map(df, size="val", use_container_width=
```

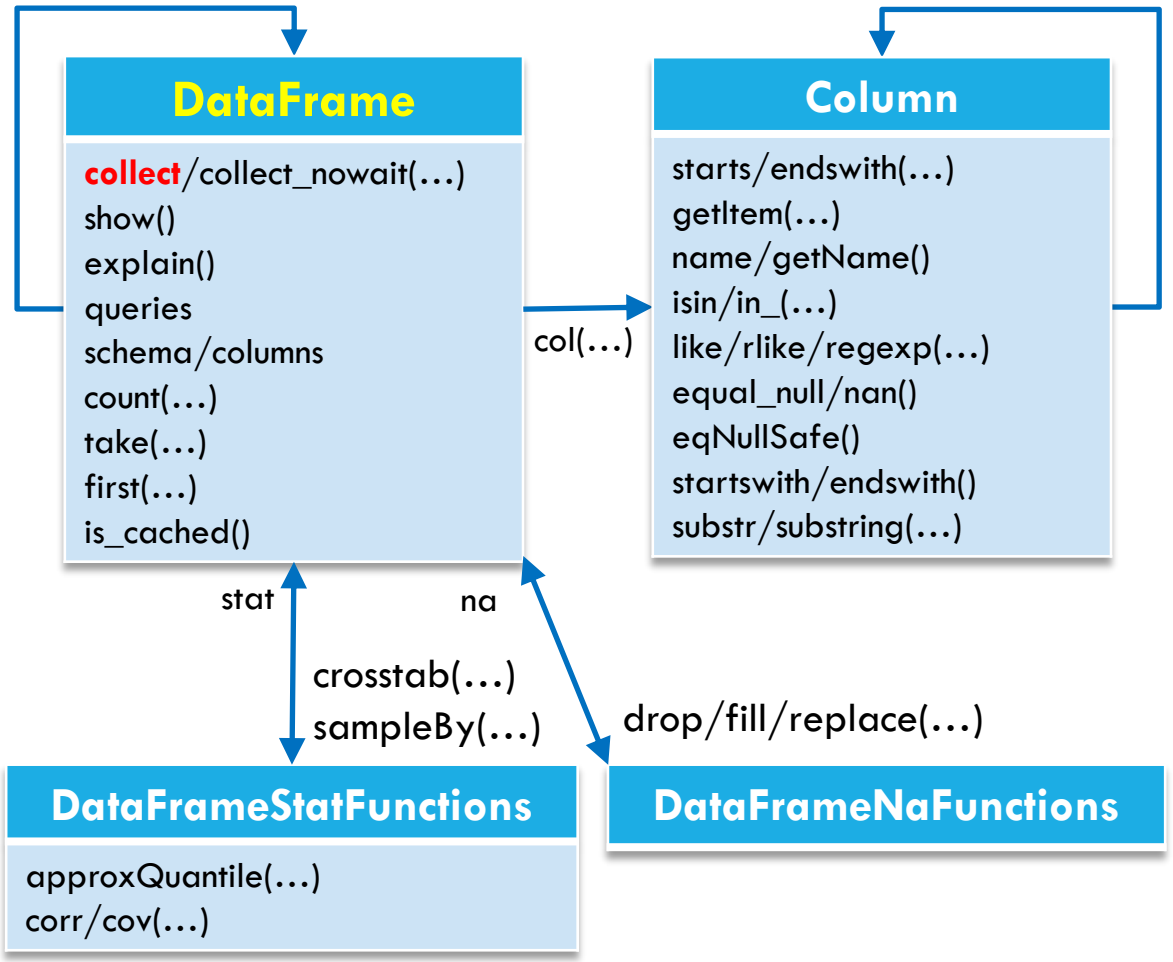


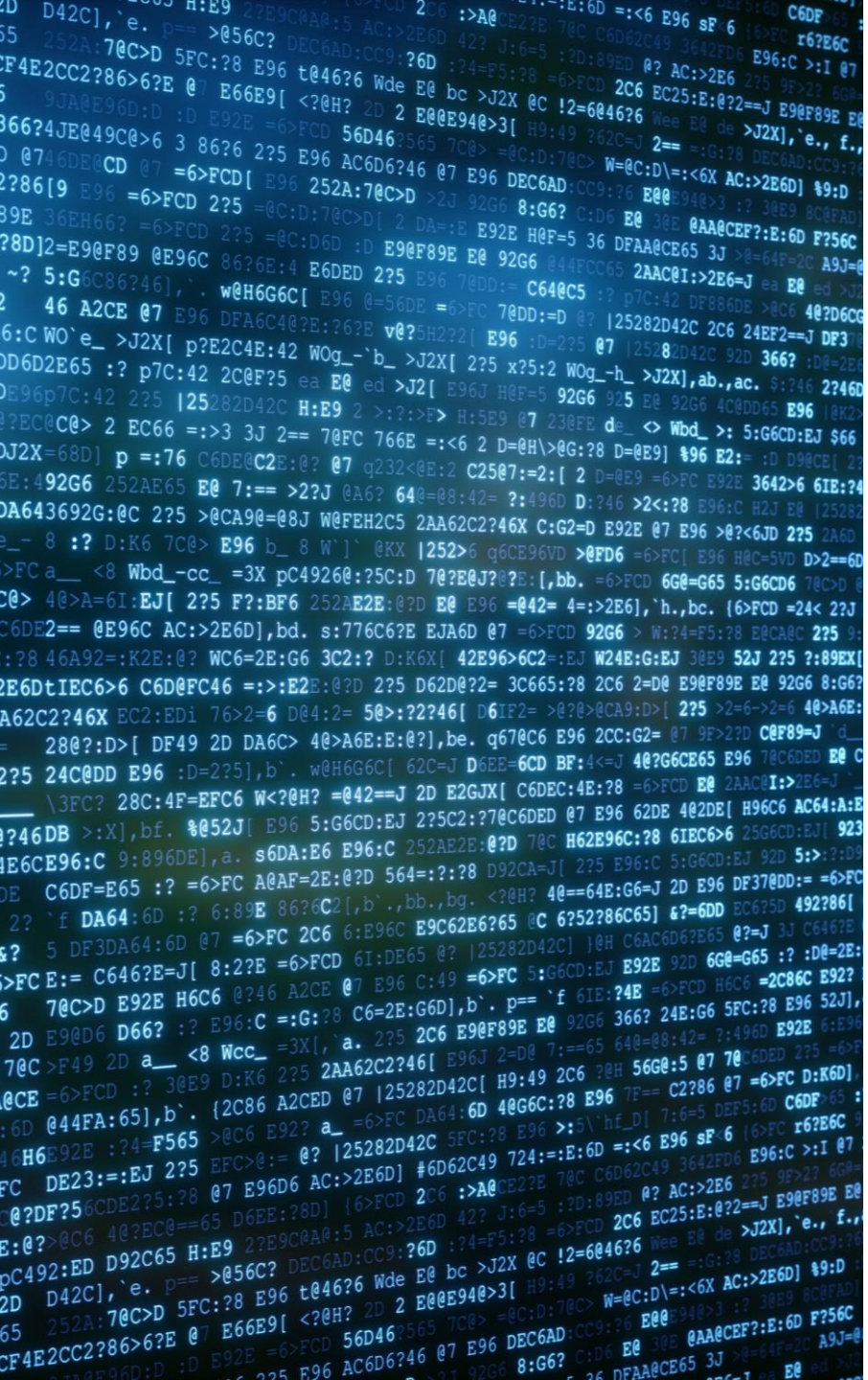
DASHBOARD IN SNOWSIGHT



SNOWPARK DATAFRAME CLASS

select/selectExpr(…)
filter/where(…)
sort/orderBy(…)
union/unionAll/unionByName(…)
intersect/except_/minus/substract
withColumn/withColumnRenamed(…)
with_columns(…)
join/crossJoin/natural_join(…)
join_table_function/flatten(…)
limit(…)
agg(…)
drop/dropna/fillna(…)
crosstab/unpivot(…)
describe/rename/replace(…)
sample/sampleBy(…)
randomSplit(…)
toDF(…)

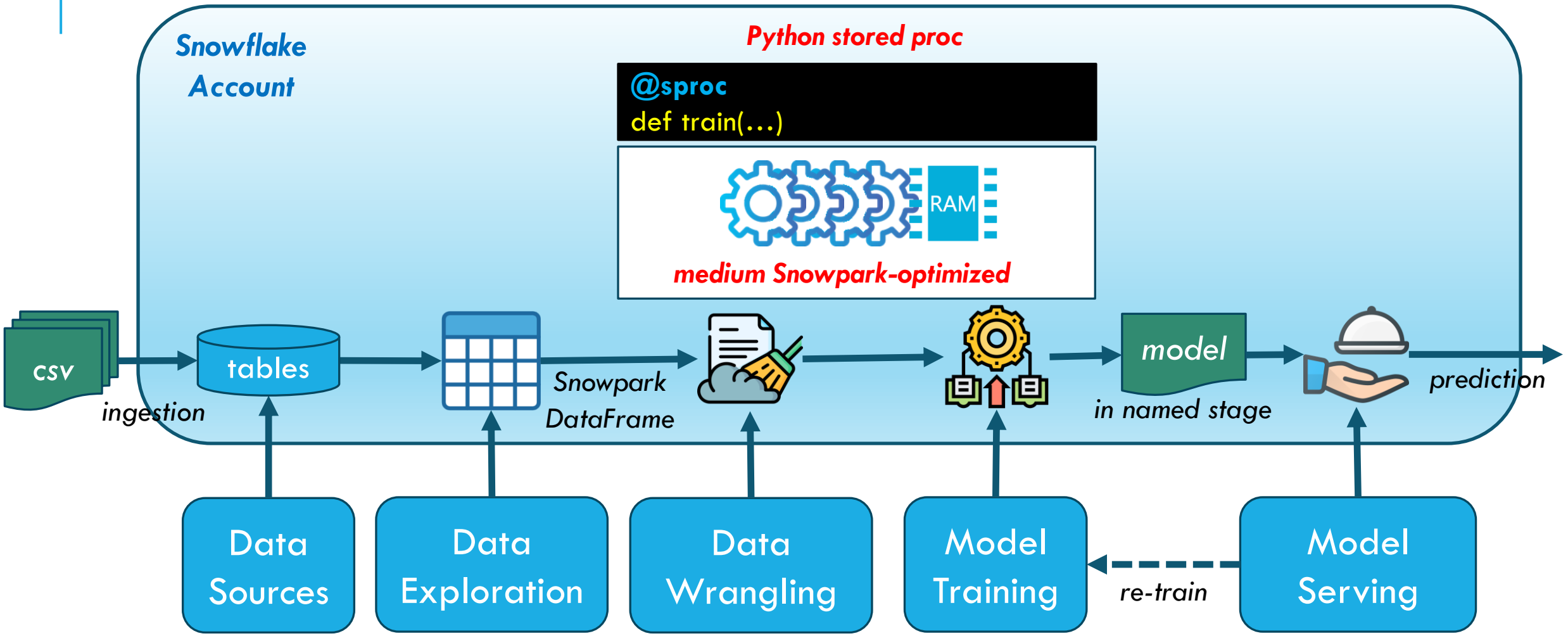


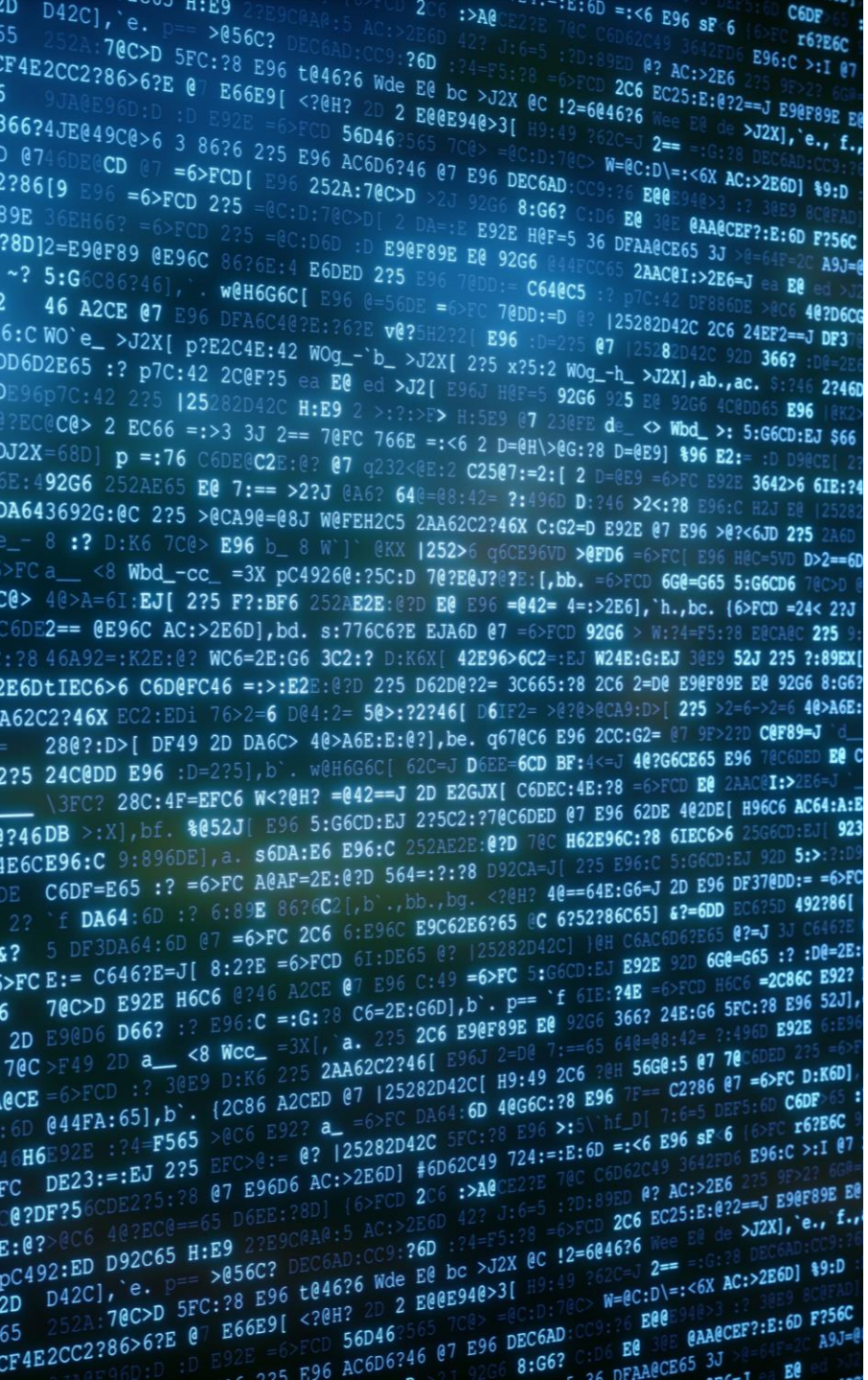


PANDAS DATAFRAME VS SNOWPARK DATAFRAME

- ***Pandas Dataframe***
  - universal popular Python library
  - ***in-memory data copy*** (on the client machine)
  - ***all processing is in-memory*** (~own SQL engine!)
  - may require a lot of memory
  - still used for Vectorized UDTFs (for I/O)
- ***Snowpark DataFrame***
  - leaves data in the database (Snowflake only)
  - ***lazy execution*** (~PySpark data frames)
  - keeps only a pointer to stored table data
  - generates/execs SQL query on action methods
  - may still require a Snowpark-optimized warehouse

ML PIPELINES USING SNOWPARK (NO CORTEX)

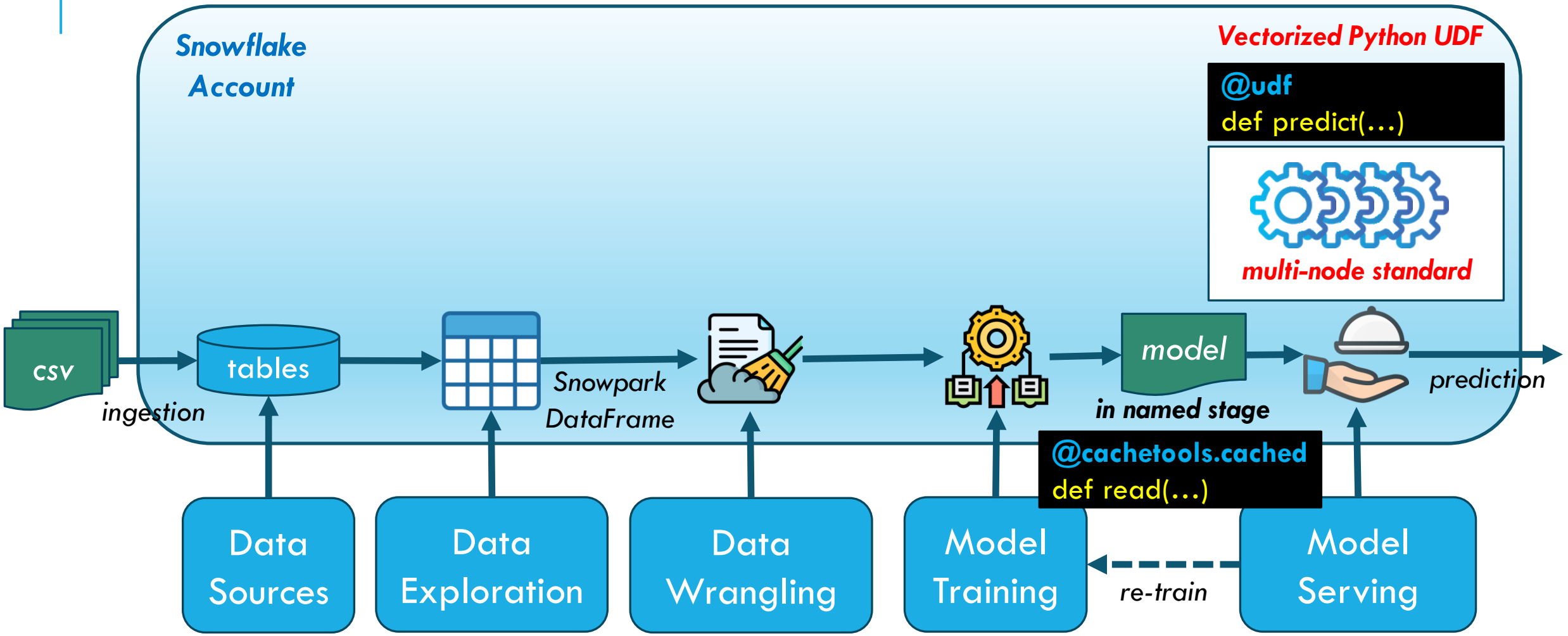


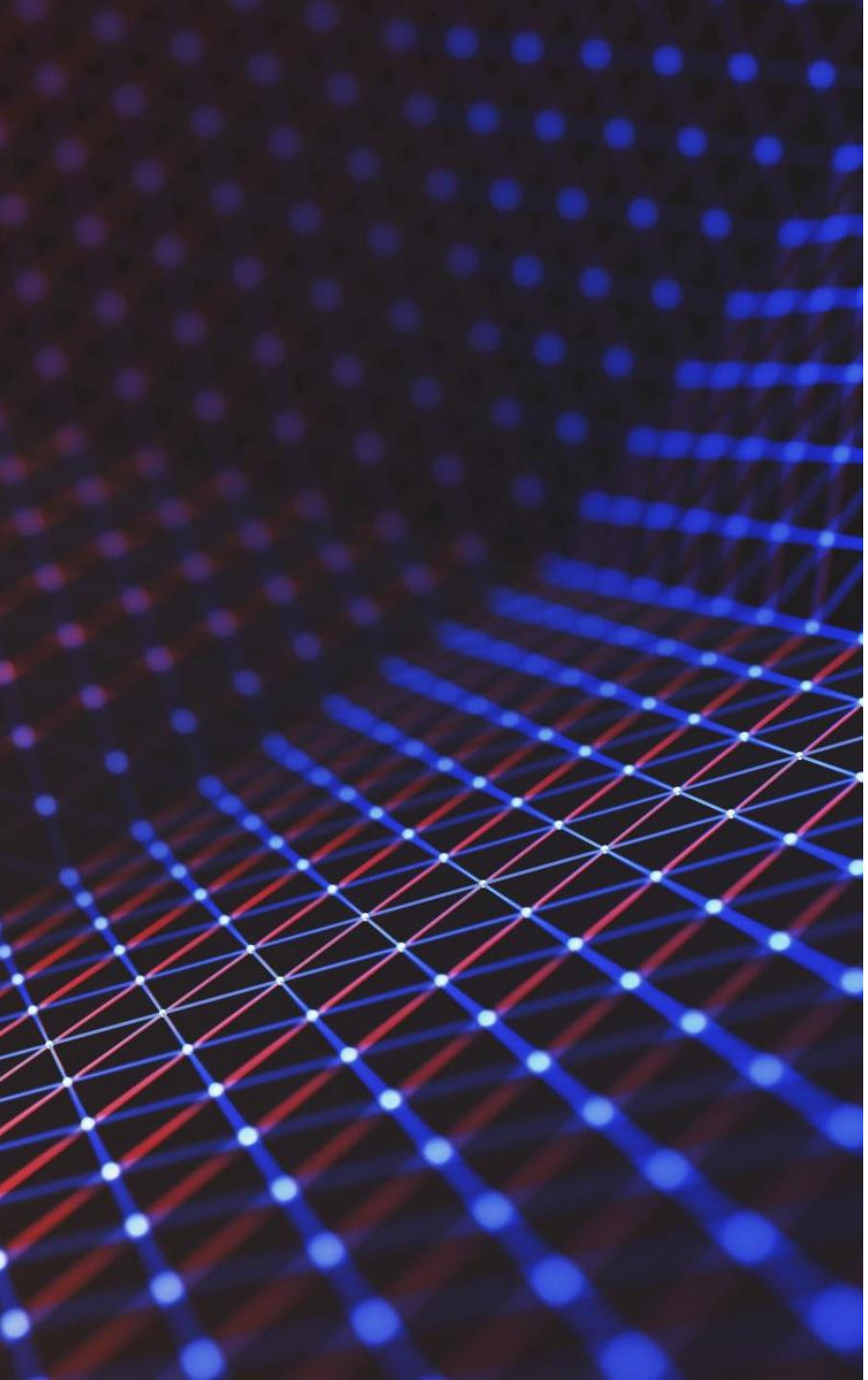


SNOWPARK PIPELINE FOR ML

- ***Snowpark DataFrame*** = faster than *Pandas DataFrame* (which is in-memory)
- ***Model Training*** = w/ Snowpark **stored-procedure** + *medium Snowpark-optimized warehouse*
- ***imports/packages*** = upload Python source files in named stage + ref w/ *add_imports/packages*(...)
- ***model*** = saved in a **named stage,** in binary Pickle format. Use ***Cachetools*** Python library to serve the model.
- ***Model Deployment*** = w/ Snowpark **vectorized UDF** + *multi-node standard warehouse*

ML PIPELINES USING SNOWPARK (NO CORTEX)

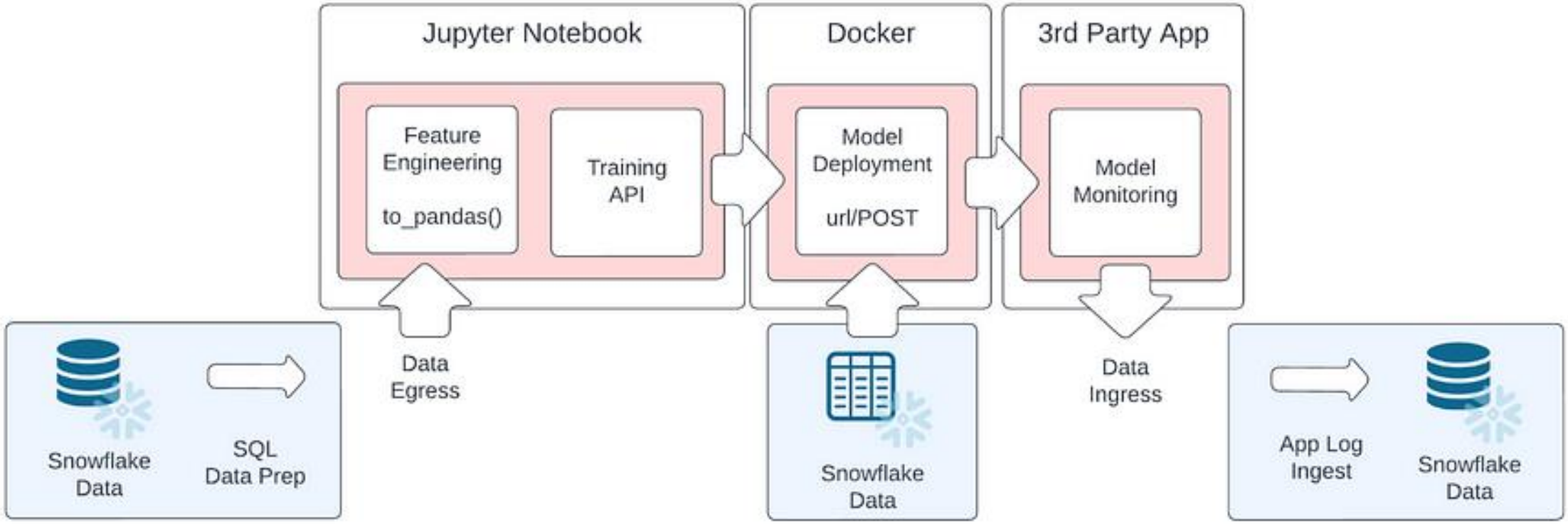




VECTORIZED PYTHON UDFS

- row-by-row processing → batches of rows (better performance!)
- **non-vectorized UDF** = scalars → UDF → scalar return
- **vectorized UDF** = scalars → ***pandas DataFrames*** → UDF → ***pandas arrays/Series*** → scalar returns

- **@vectorized(input=pandas.DataFrame)**
- optional ***max_batch_size***

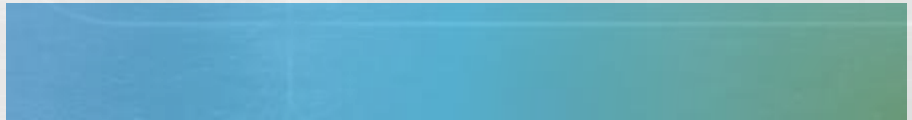

Jupyter Notebook
Feature Engineering
to_pandas()
Training API
Docker
Model Deployment
url/POST
3rd Party App
Model Monitoring
Snowflake Data
SQL Data Prep
Data Egress
Snowflake Data
Data Ingress
App Log Ingest
Snowflake Data

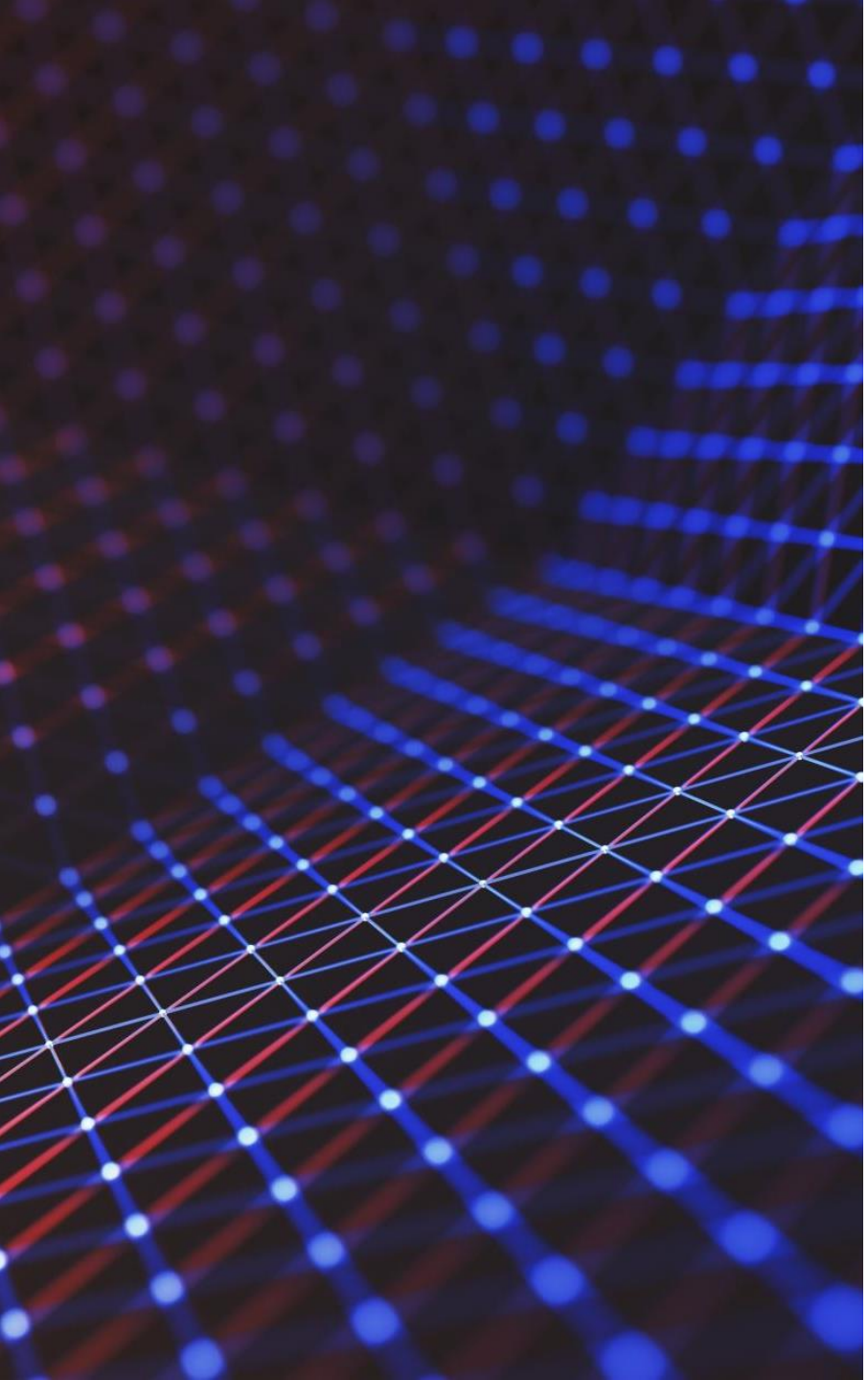
SNOWPARK ML APIS

Modeling

Operations (MLOps)

Data Access

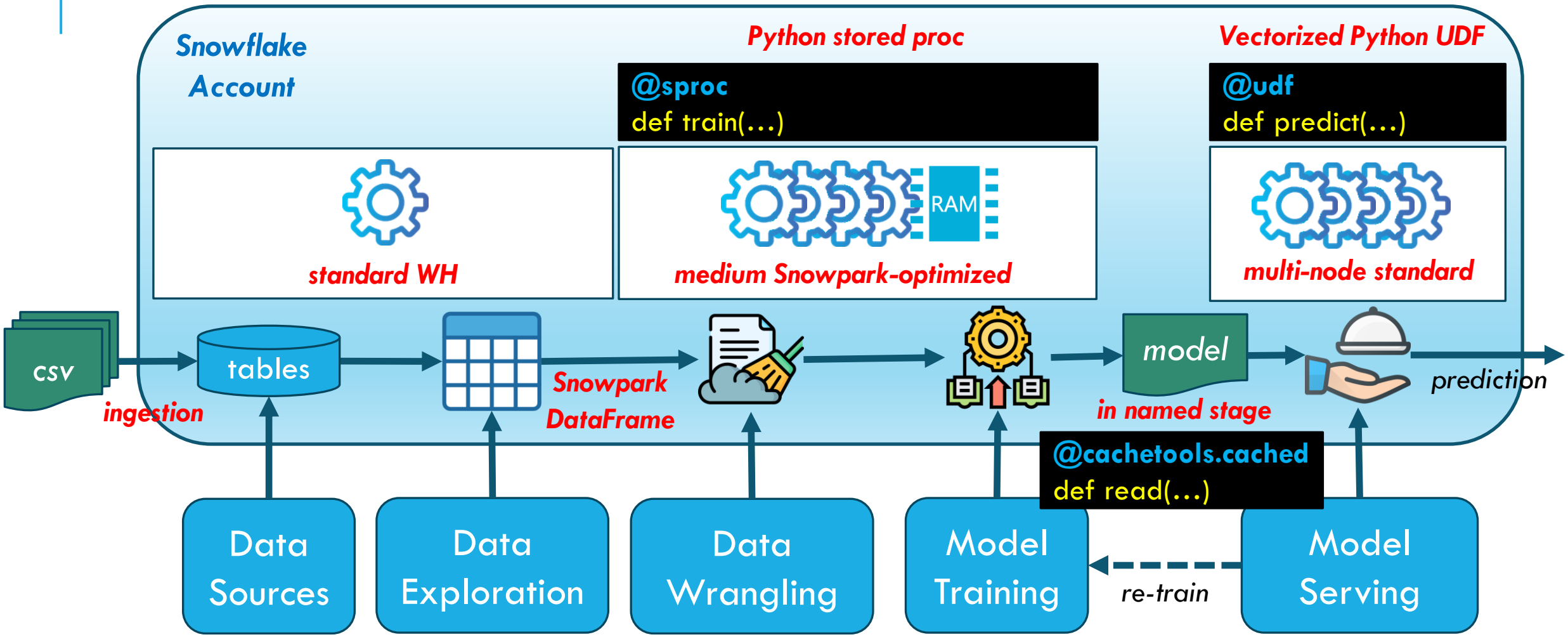




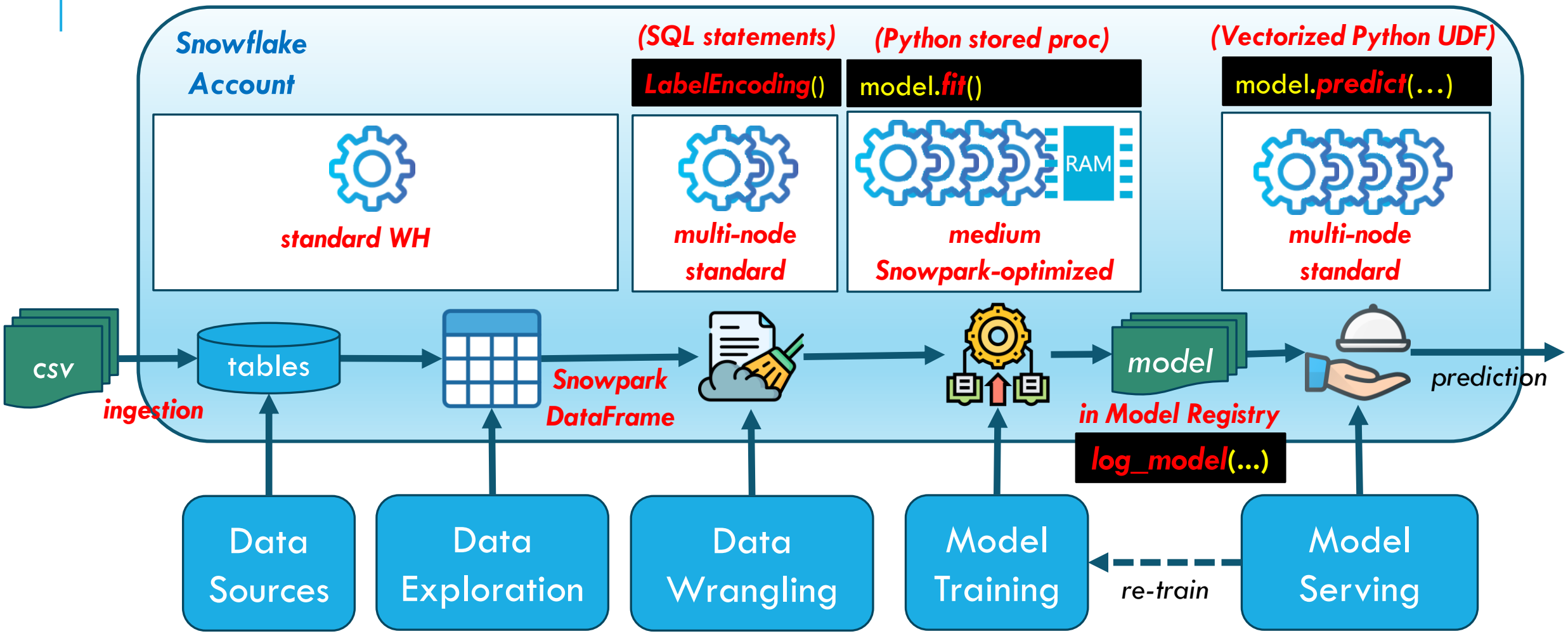
SNOWPARK ML

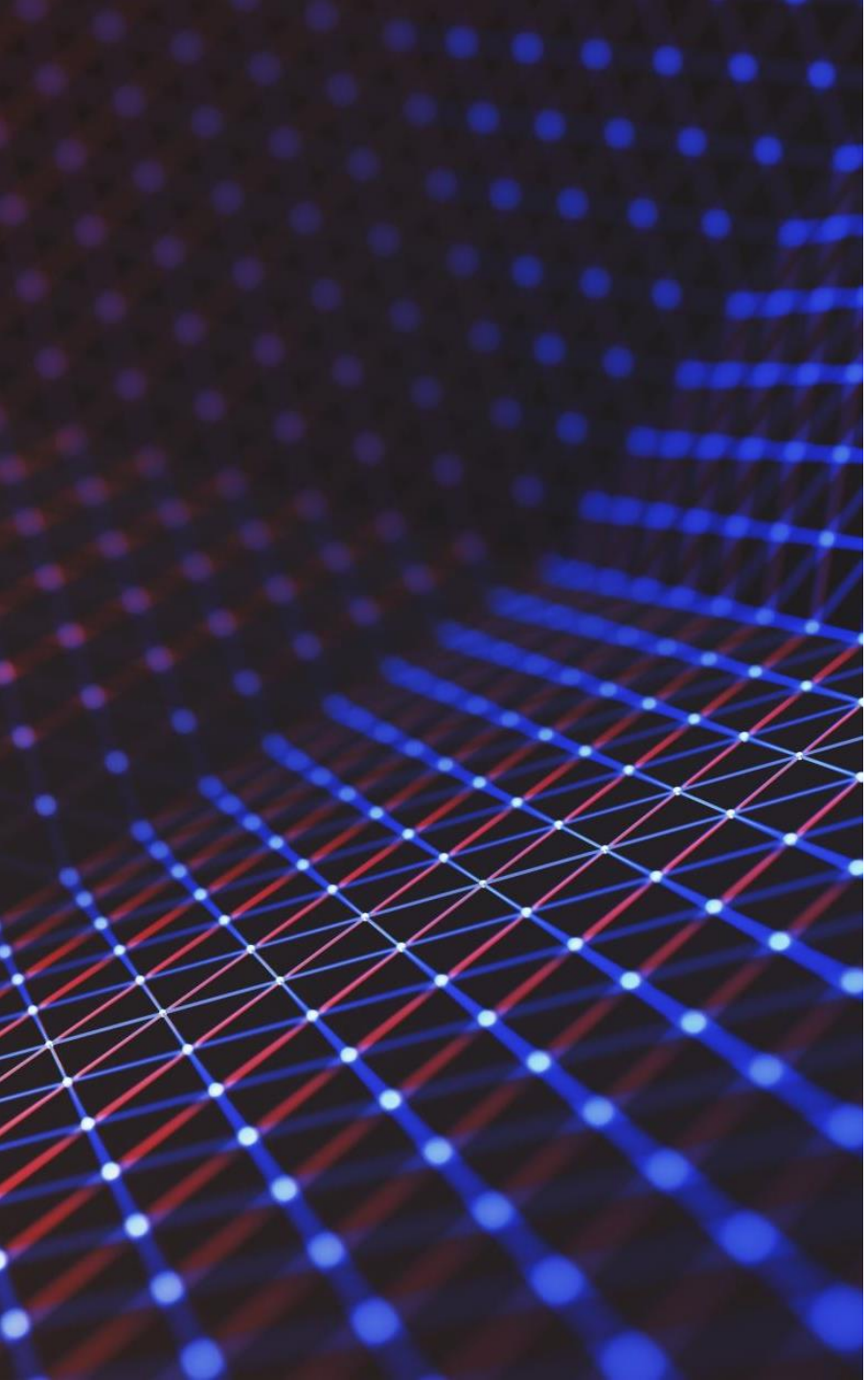
- **Modeling**
  - wrappers around **scikit-learn/XGBoost/LightGBM**
  - ***fit/transform()*** distributed SQL preprocessing
  - ***fit()*** automatic stored proc for model training
  - distributed HPO, w/ **Grid/RandomizedSearchCV**
  - distributed performance metrics
- **Operations (MLOps)**
  - ***predict()*** model serving in temp vectorized UDF
  - **Model Registry**
- **Data Access**
  - **FileSystem/FileSet** APIs
  - PyTorch/TensorFlow **Framework Connectors**

ML PIPELINES USING SNOWPARK (NO CORTEX)



ML PIPELINES WITH SNOWPARK ML (IN CORTEX)

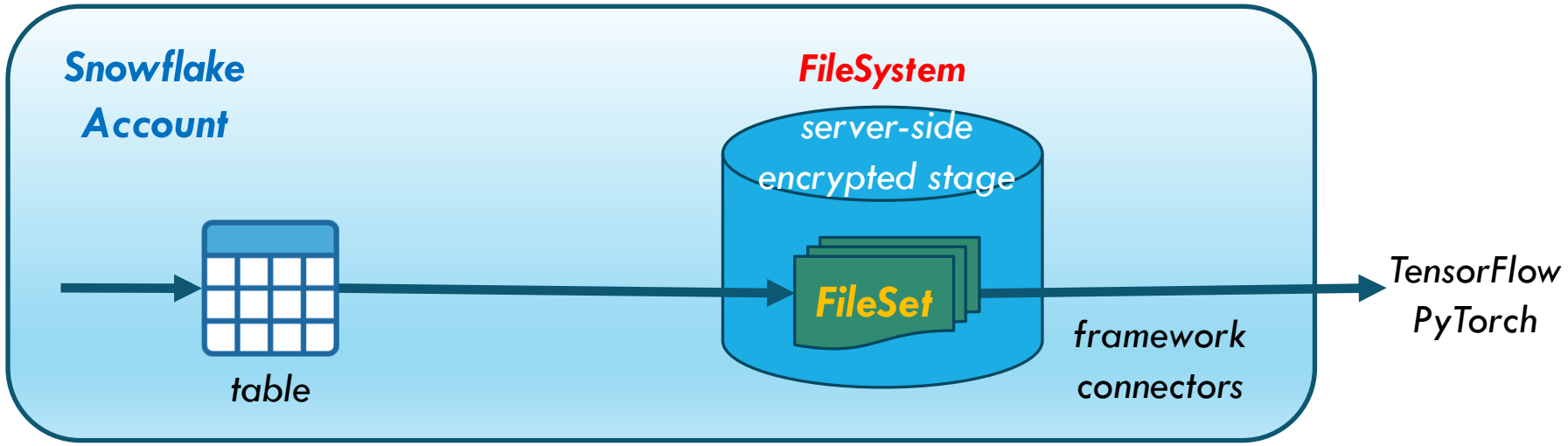




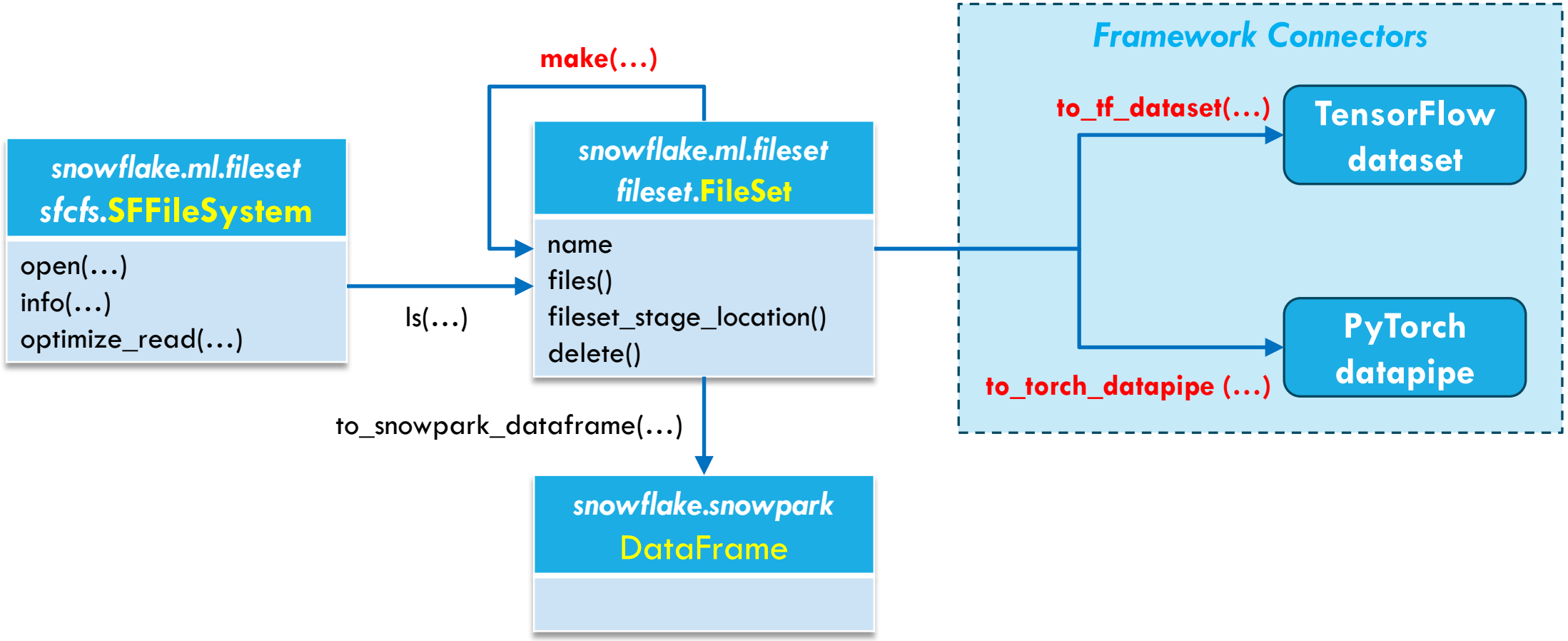
SNOWPARK ML

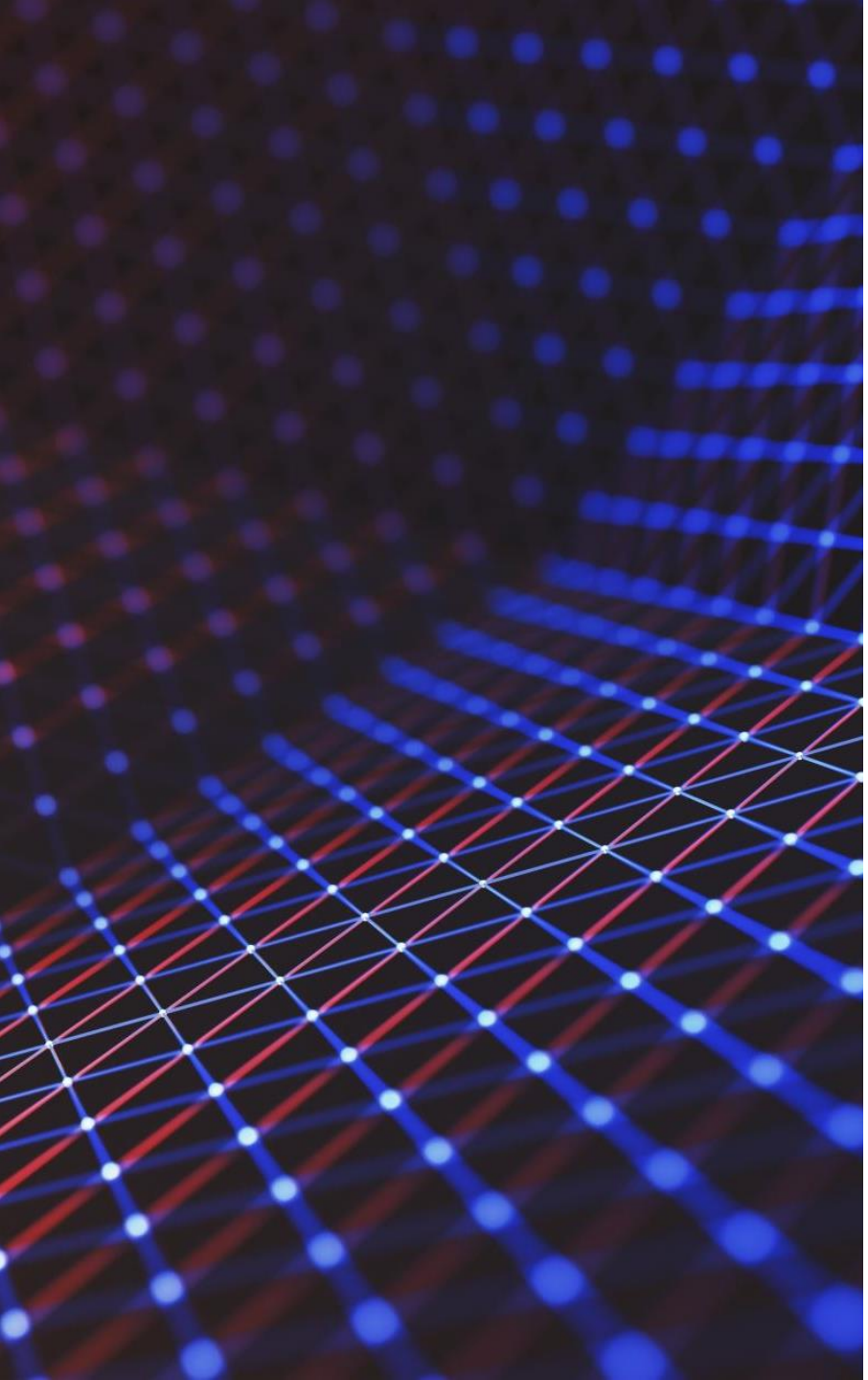
- **ctors** (for estimators/transformers) → w/ 5 additional (optional) params:
  - **input/output_cols** = col names
  - **sample_weight_col/label_cols** = in a Snowpark/Pandas DataFrame
  - **drop_input_cols** = True to filter for output cols
- ***fit/predict()*** → w/ a Snowpark/Pandas DataFrame (not inputs/weights/labels arrays)
- ***transform/predict()*** → return DataFrame w/ all cols from input DataFrame (not arrays)
- ***fit_transform()*** → not in transformers! chain rather w/ .fit(df).transform(df)
- ***inverse_transform()*** → not in transformers (unnecessary)

SNOWPARK ML DATA ACCESS



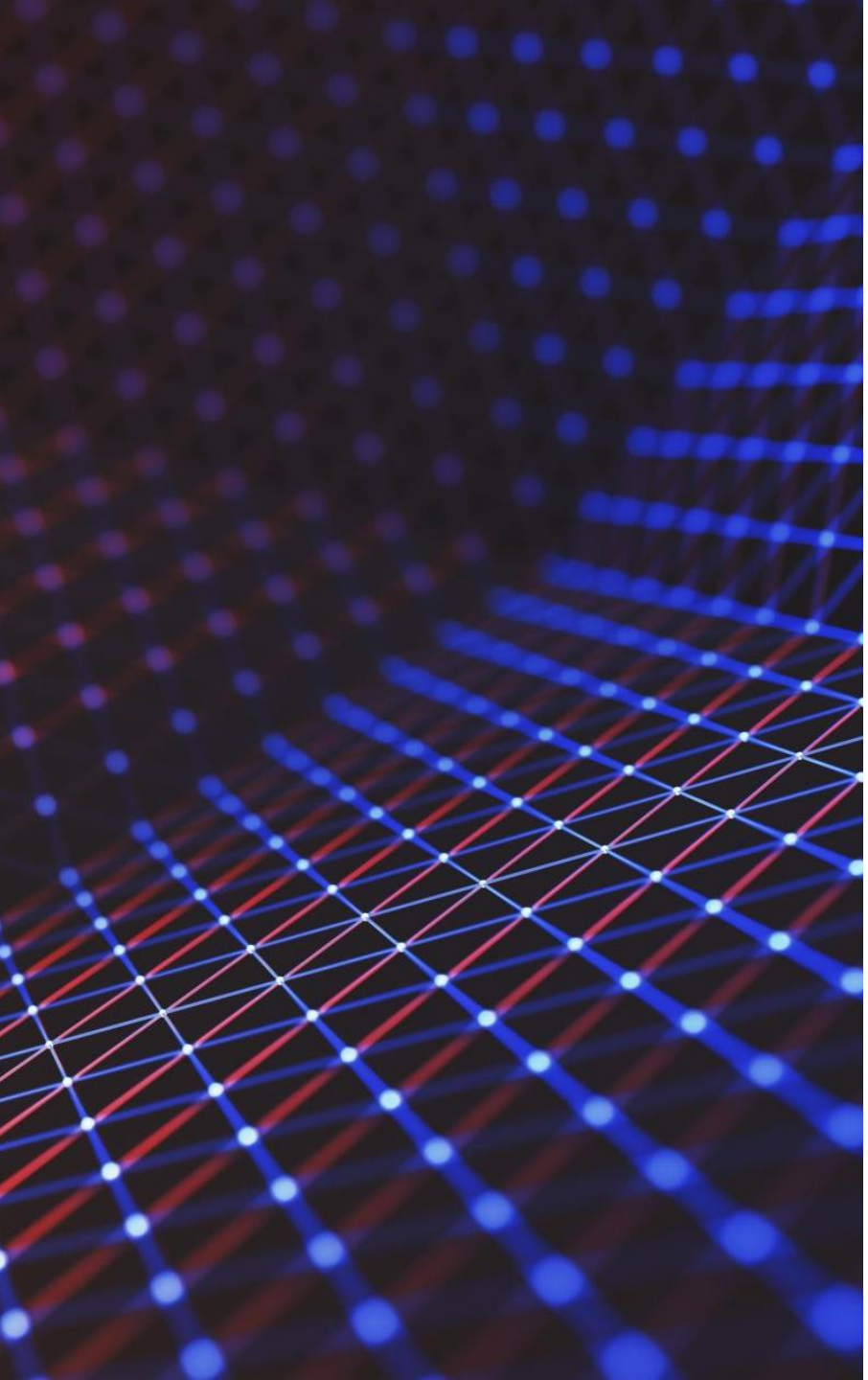
SNOWPARK ML DATA ACCESS: OBJECT MODEL





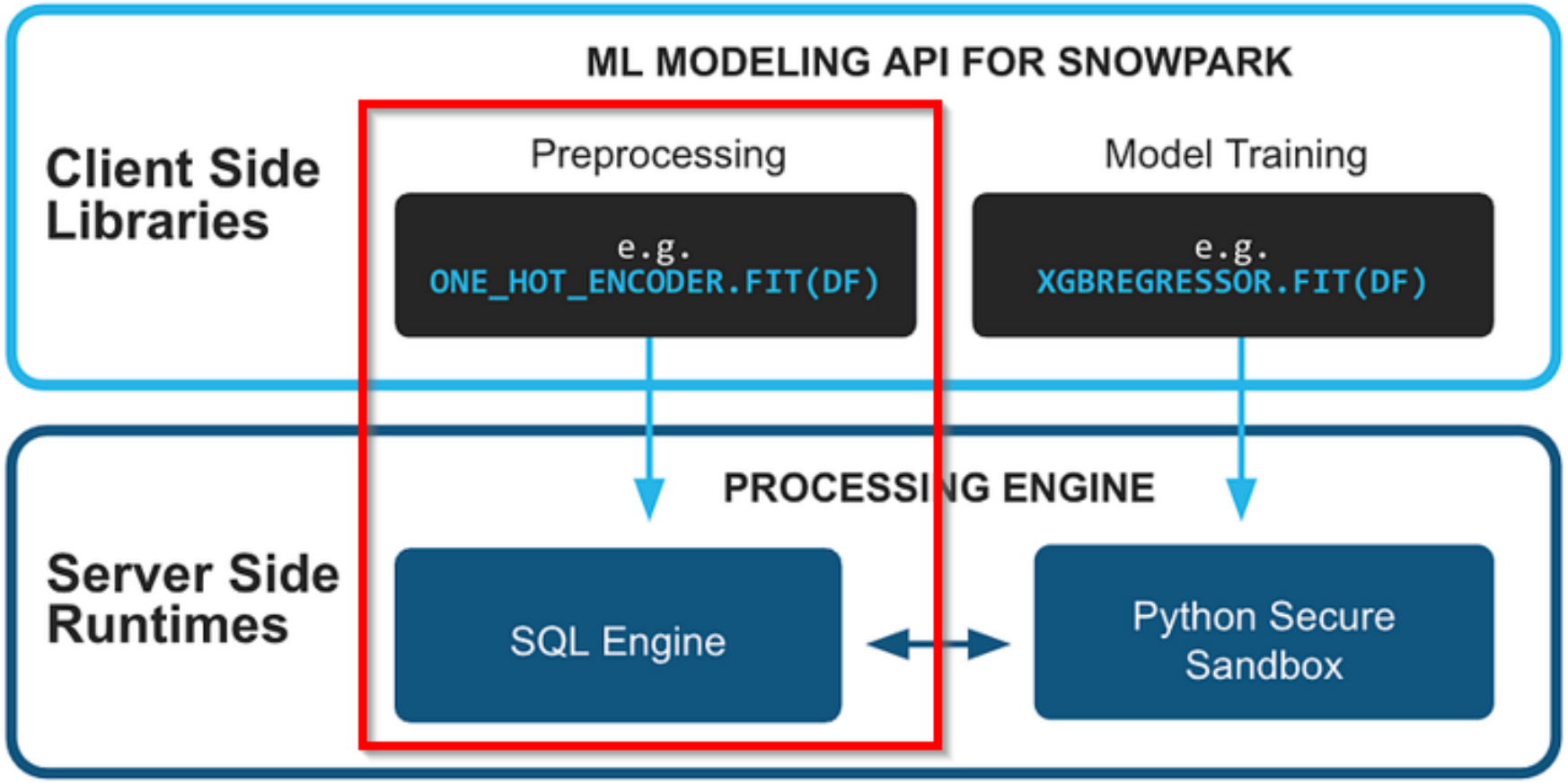
SNOWFLAKE ML DATA ACCESS

- **SFFileSystem** = to access raw content of a staged file from client code (in ***server-side encrypted stage*** only!)
- **FileSet** = persisted snapshot of a query result or table content
- ***framework connectors*** = FileSet → **PyTorch/TensorFlow** formats (tensors/records)

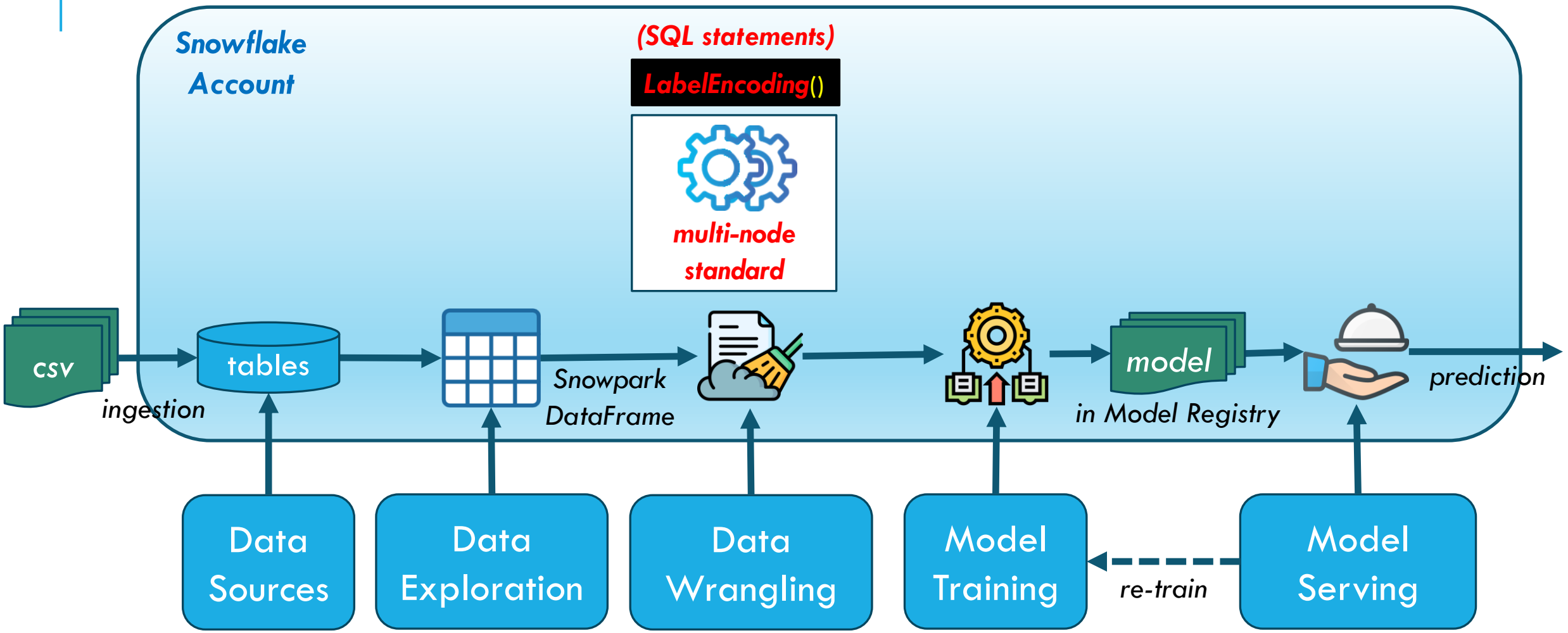


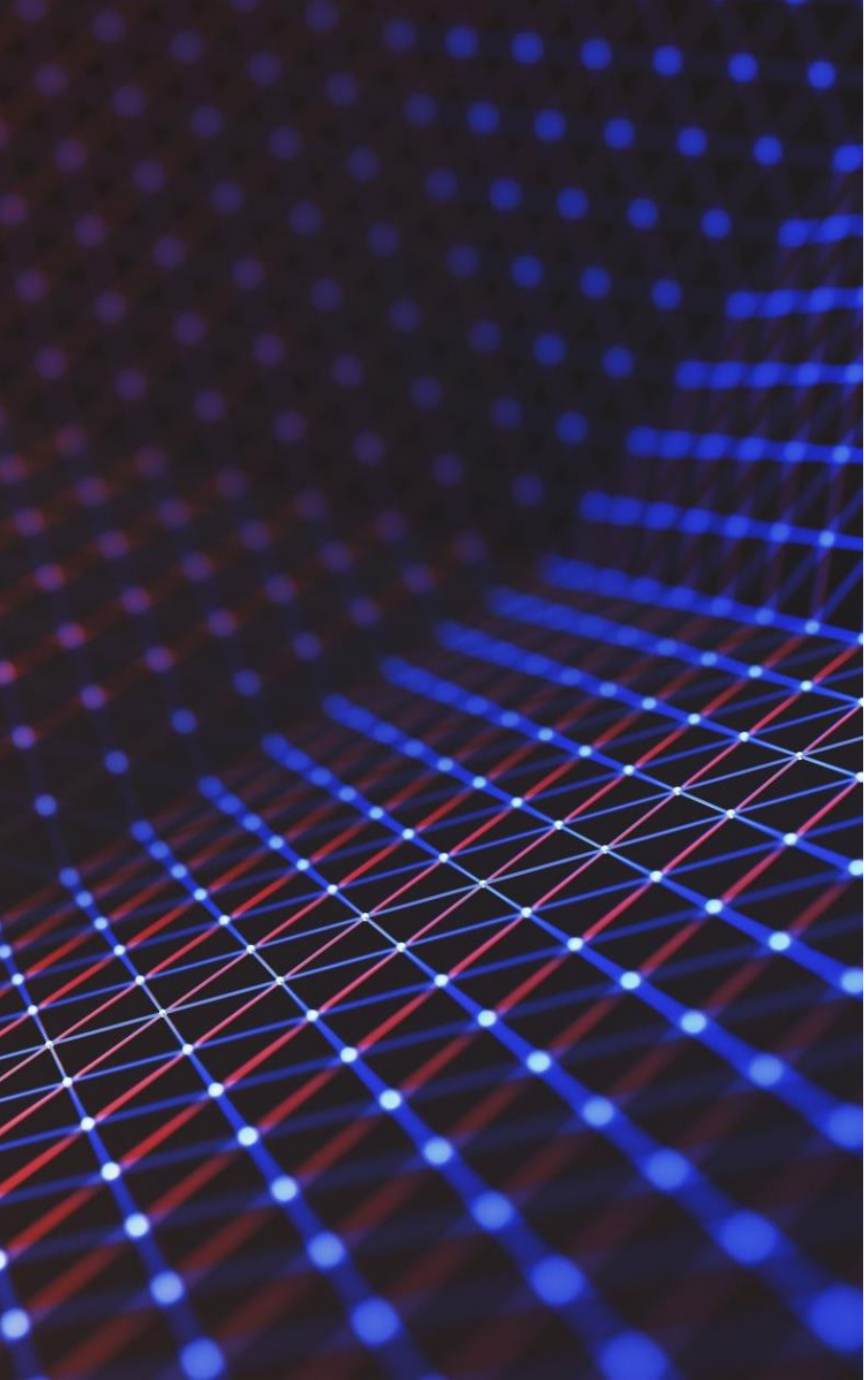
SNOWFLAKE
STAGED FILES ACCESS

- upload/unload files ← **PUT/GET**
- staged file maniputation ← **LIST/REMOVE**
- table to/from files ← **COPY INTO**
- query staged files ← $1, $2...
- staged files structure ← external tables, table directories
- staged files accessed from a UDF ← session.***add_import***(...)
- content of staged files from a UDF ← ***SnowflakeFile***
- cached content of model files ← ***cachetools***


ML MODELING API FOR SNOWPARK
Client Side Libraries
Preprocessing
e.g.
ONE_HOT_ENCODER.FIT(DF)
Model Training
e.g.
XGBREGRESSOR.FIT(DF)
PROCESSING ENGINE
Server Side Runtimes
SQL Engine
Python Secure Sandbox

ML PIPELINES WITH SNOWPARK ML (IN CORTEX)

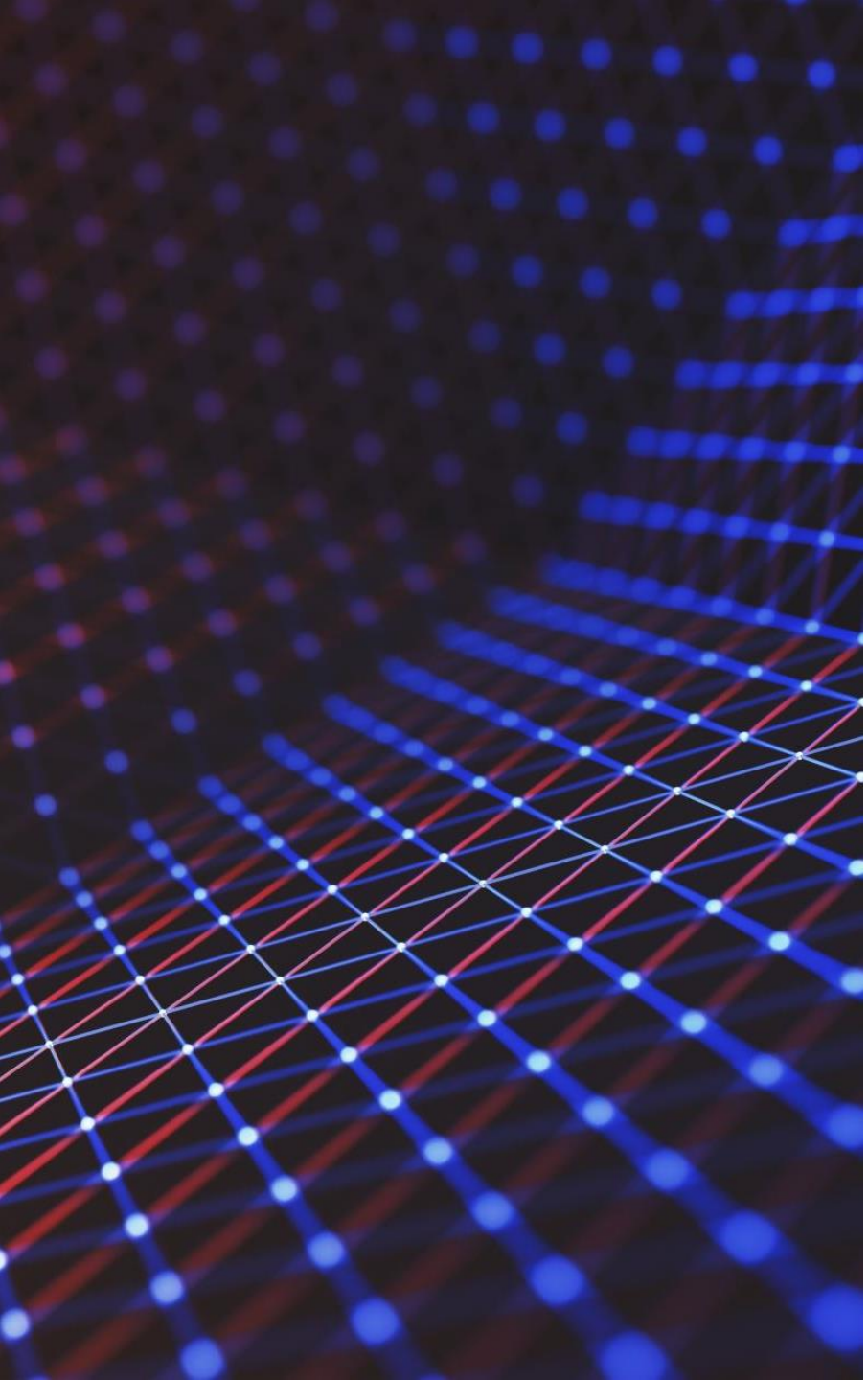




TRANSFORMER CLASSES

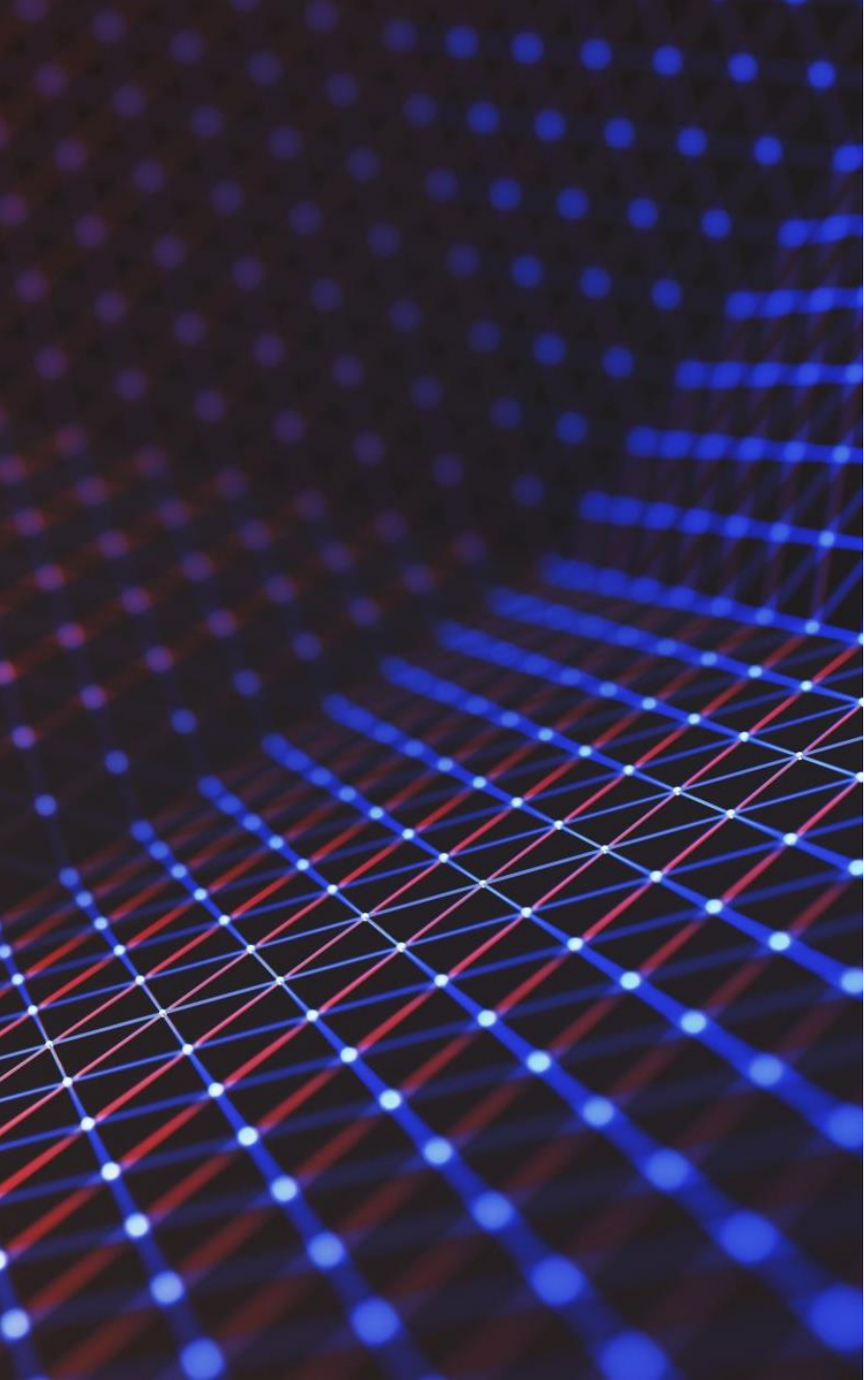
- Normalizer
- Standard/MinMax/MaxAbs/RobustScaler
- OneHot/Ordinal/LabelEncoder
- Binarizer, KBinsDiscretizer
- PolynomialFeatures - not distributed!

- ColumnTransformer
- Pipeline



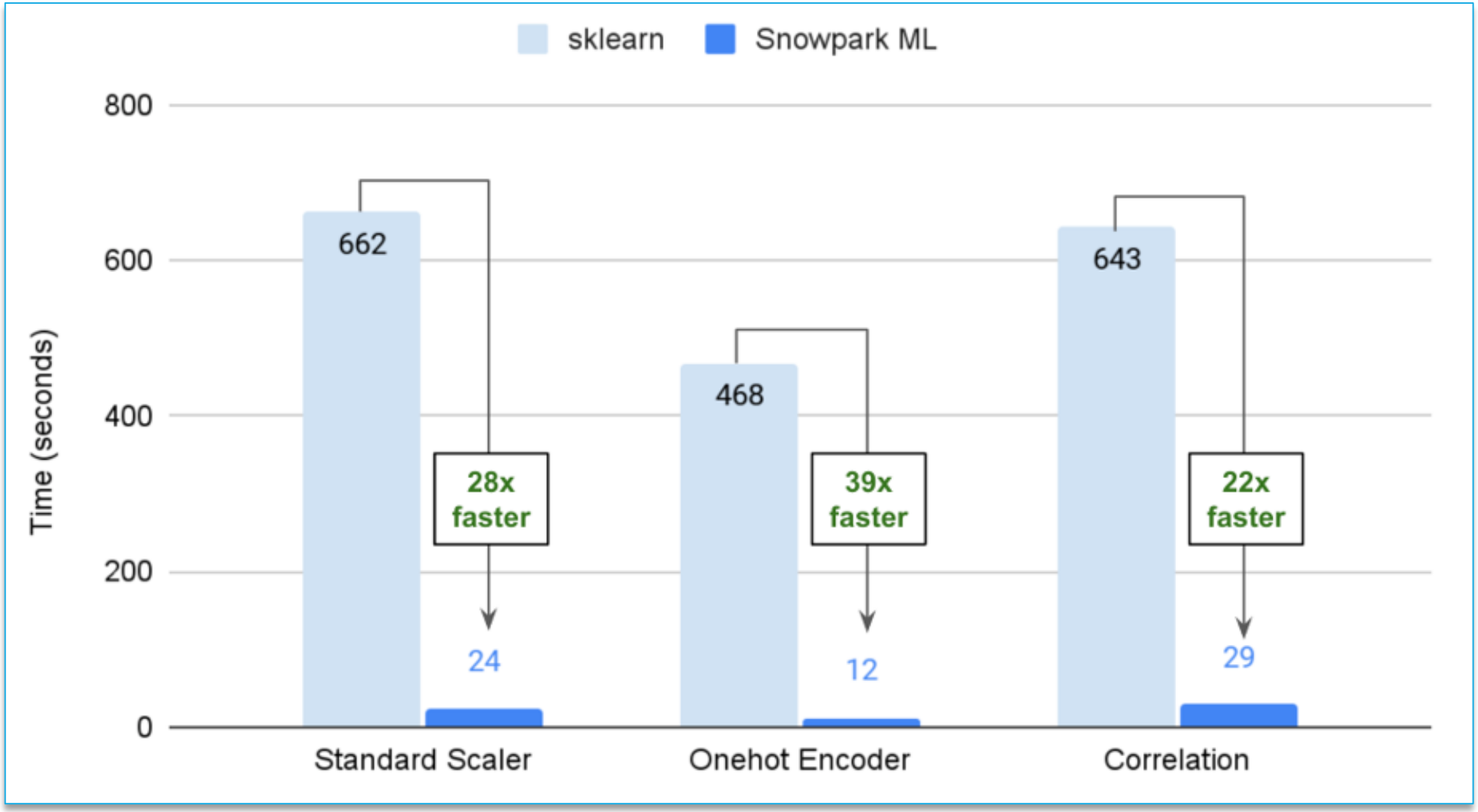
DISTRIBUTED FIT()

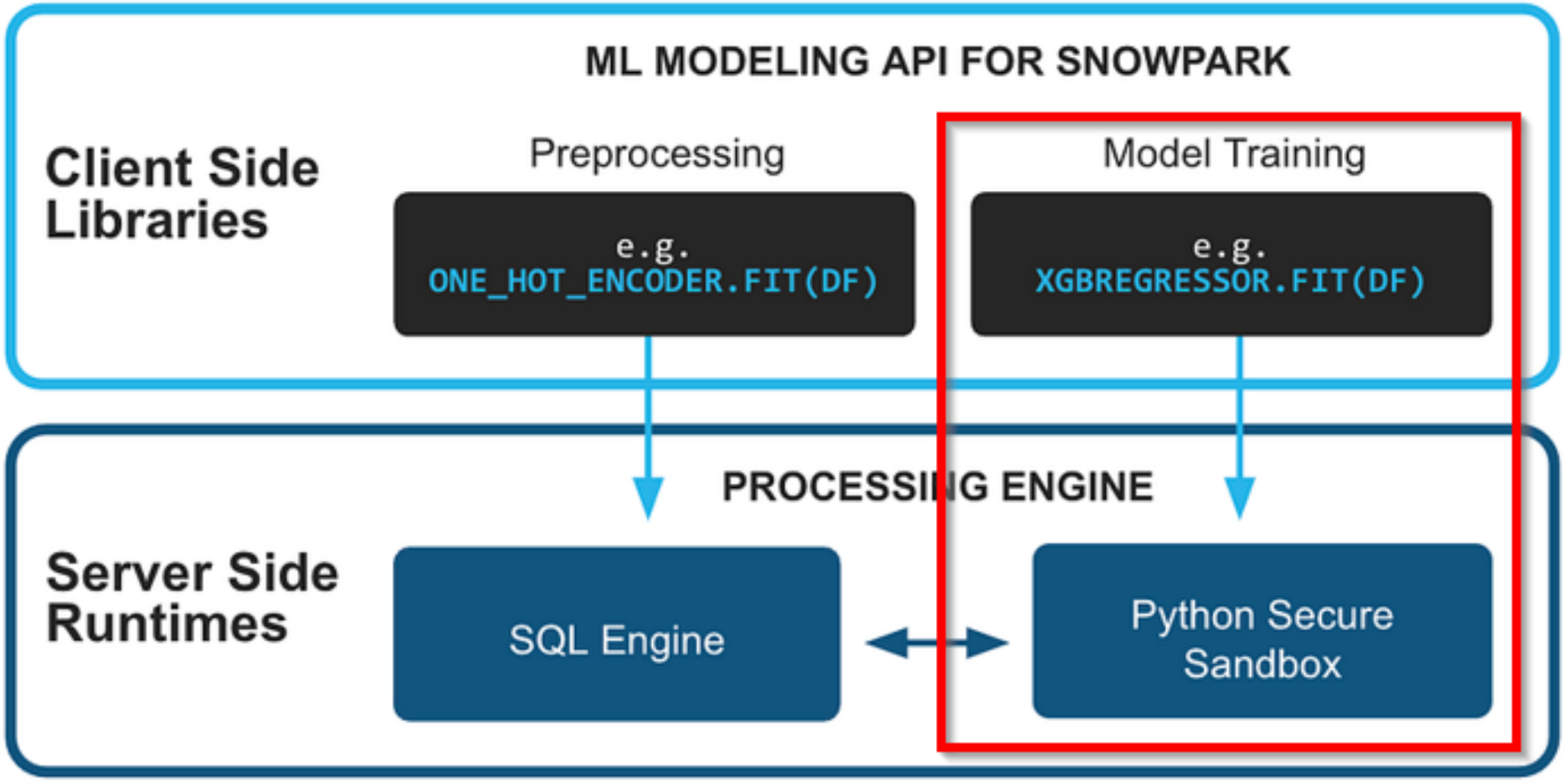
- w/ Snowpark DataFrames → distrib SQL engine, in parallel, 25..50x faster (w/ pandas DataFrames → fitted locally, ~scikit-learn)
- transformers generate SQL queries for mean/max/count… → result materialized locally (or w/ intermediate results + local computations over metadata)
- complex transformers that req temp state tables (ex OneHotEncoder, OrdinalEncoder) → tables are represented locally w/ pandas DataFrames



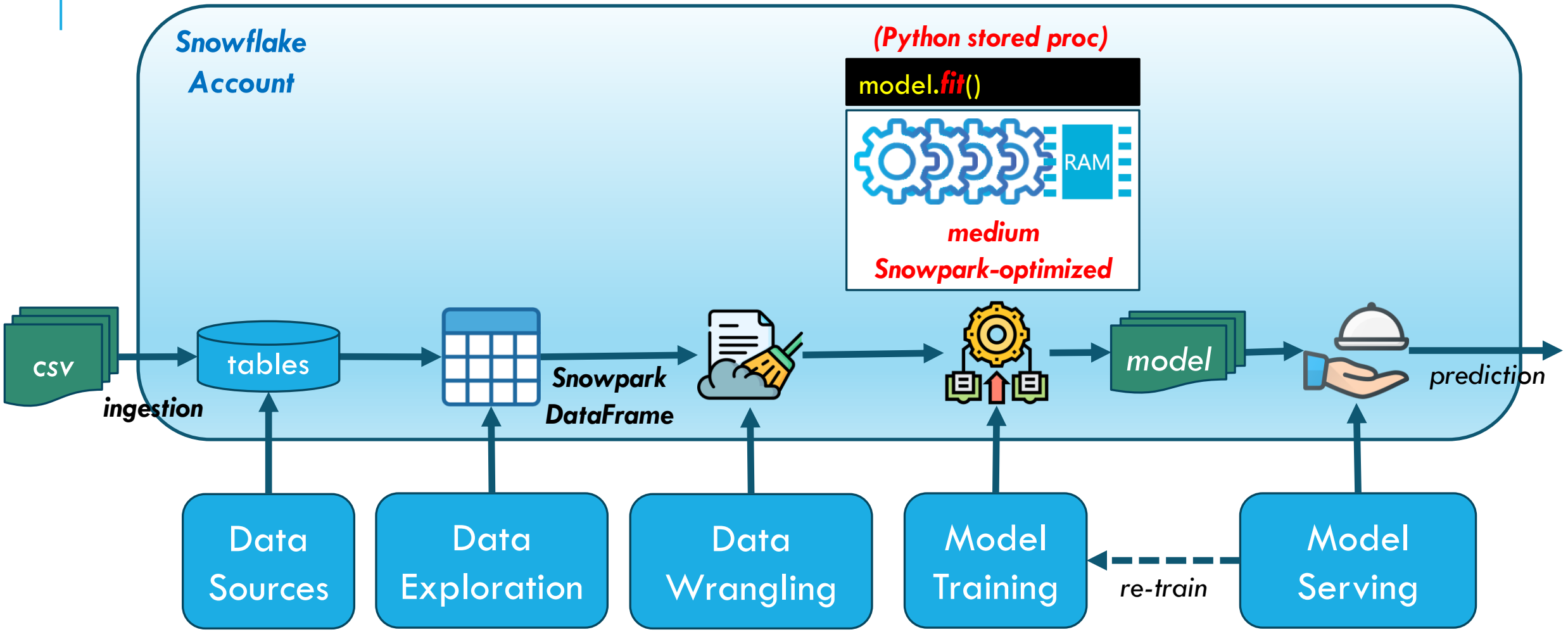
DISTRIBUTED TRANSFORM()

- fitted transformer generates a Snowpark DataFrame with underlying SQL queries representing the transformed dataset
- ***lazy evaluation*** (for StandardScaler, MinMaxScaler) → no transform is actually performed!
- certain complex transforms involve execution (for OneHotEncoder, OrdinalEncoder) → w/ temp table from Pandas DataFrame (which stores the state of the object) for joins and other operations

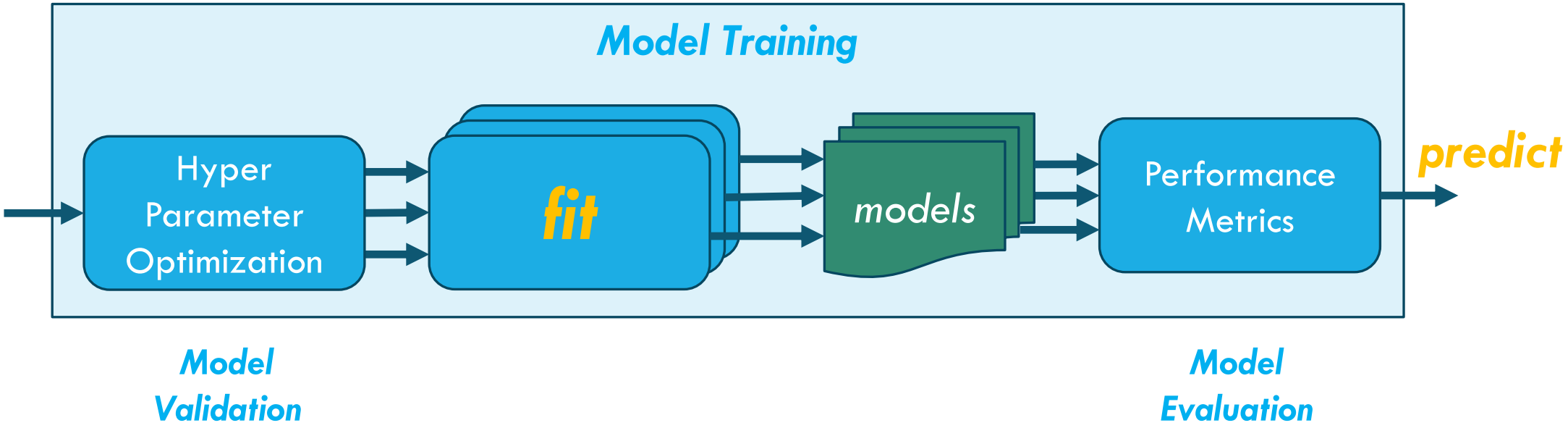

sklearn
Snowpark ML
800
600
400
200
0
Time (seconds)
662
24
28x faster
468
12
39x faster
643
29
22x faster
Standard Scaler
Onehot Encoder
Correlation

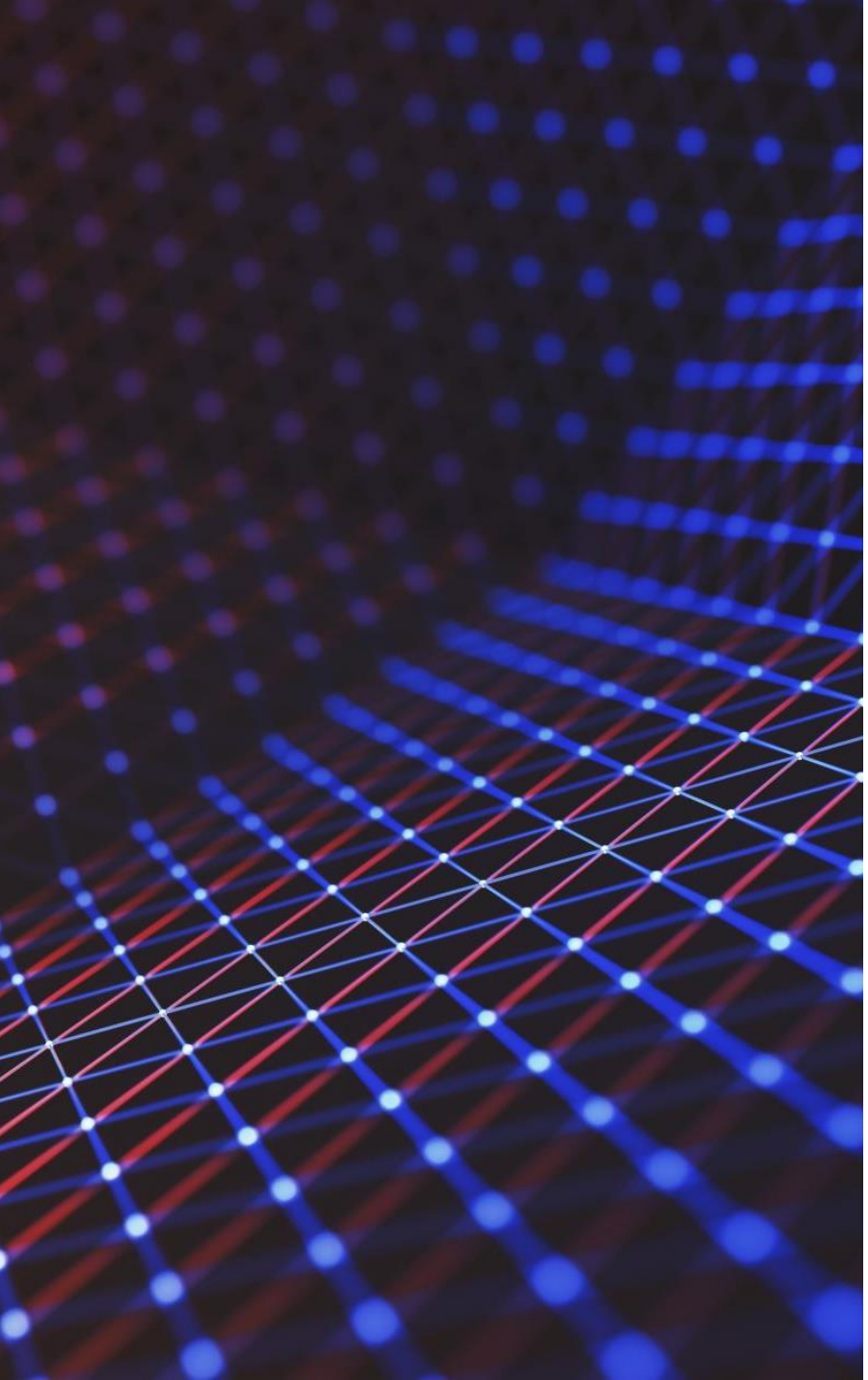

ML MODELING API FOR SNOWPARK
Client Side Libraries
Preprocessing
Model Training
e.g.
ONE_HOT_ENCODER.FIT(DF)
e.g.
XGBREGRESSOR.FIT(DF)
PROCESSING ENGINE
Server Side Runtimes
SQL Engine
Python Secure Sandbox

ML PIPELINES WITH SNOWPARK ML (IN CORTEX)



MODEL VALIDATION AND EVALUATION

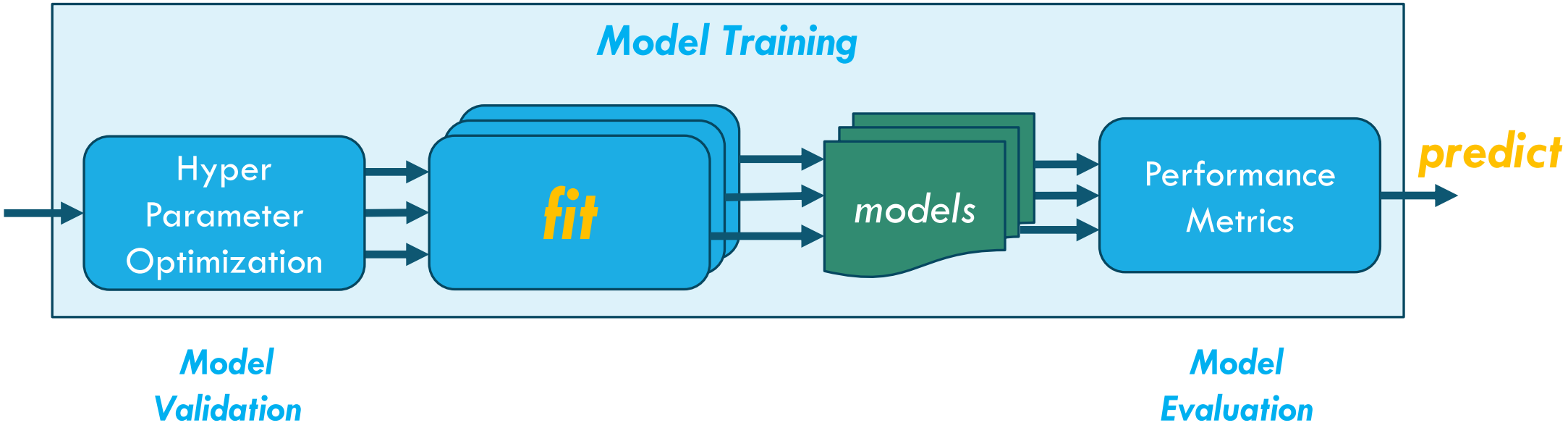




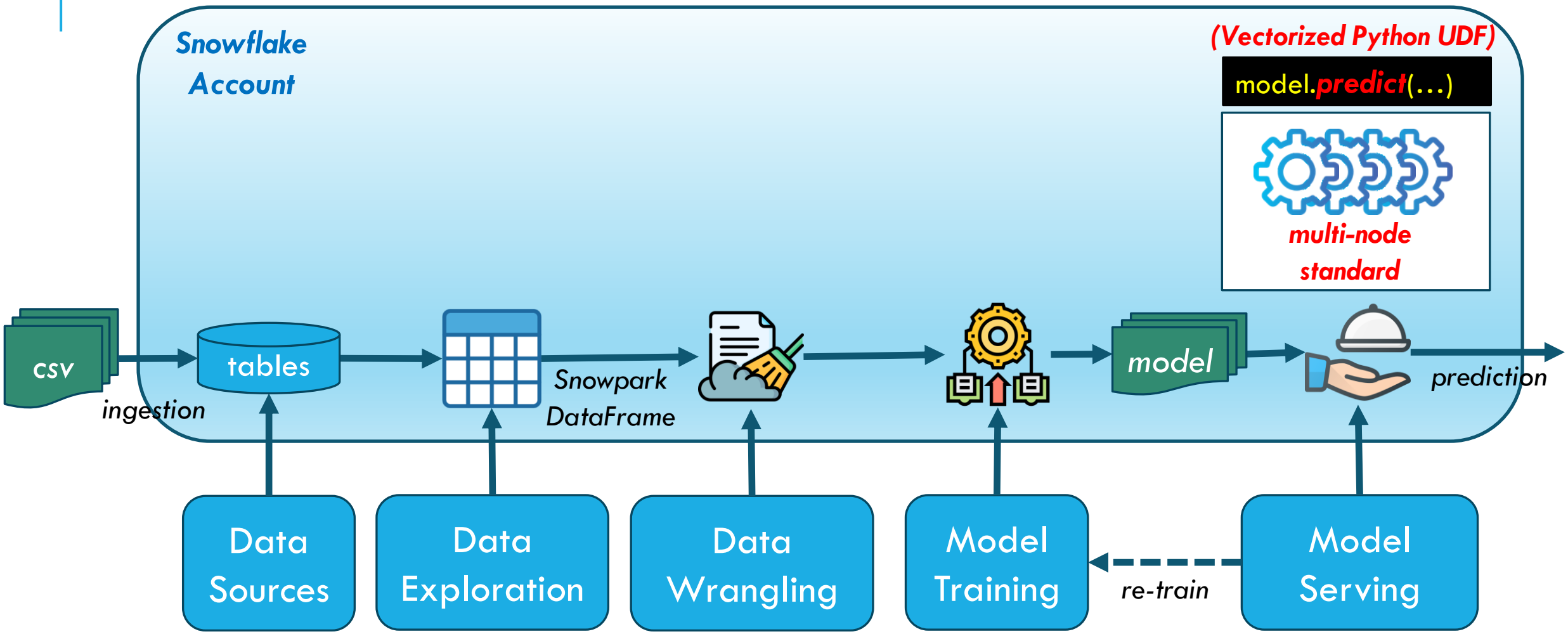
DISTRIBUTED HPO (FOR MODEL VALIDATION)

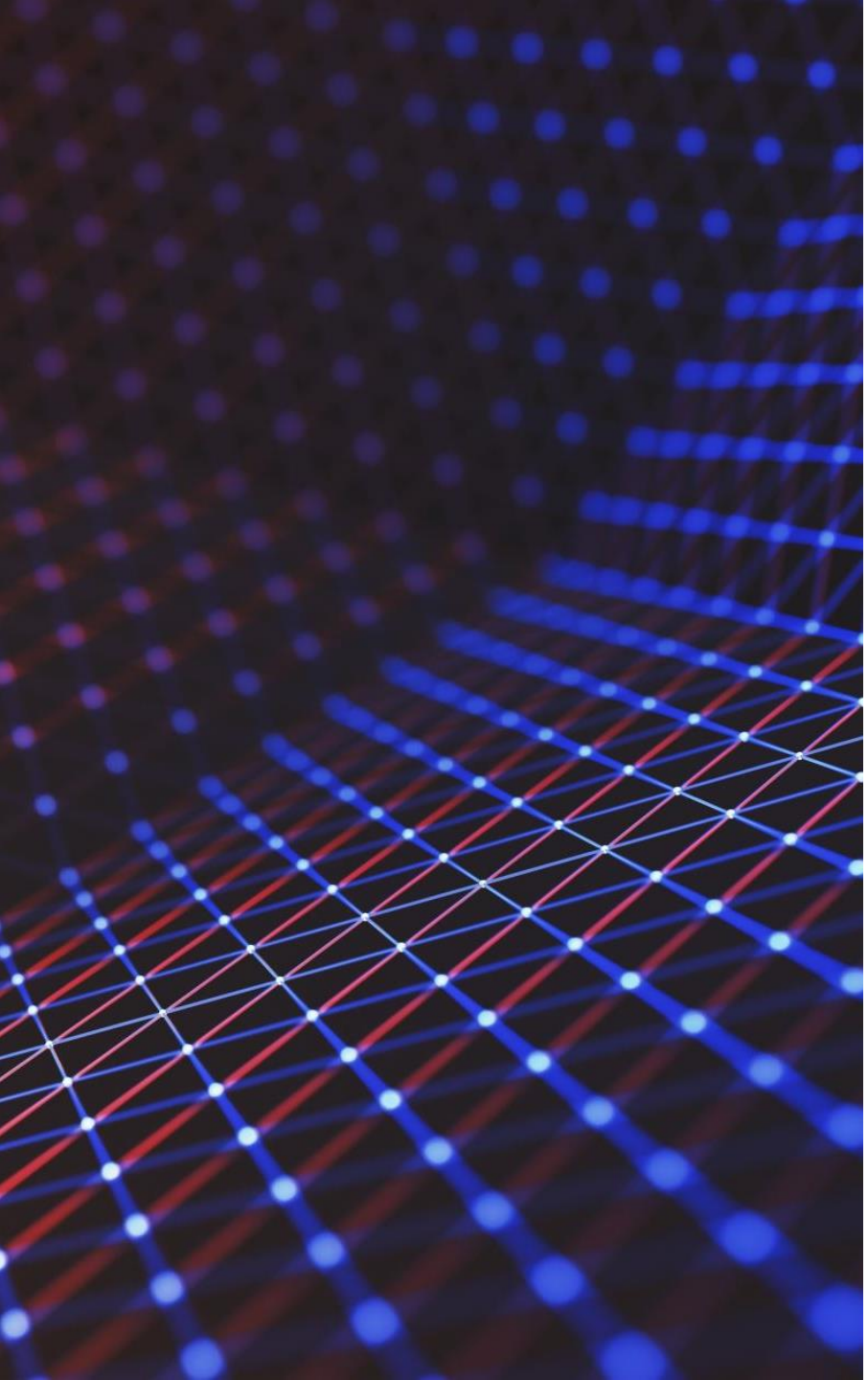
- train multiple models in parallel → on single/multiple-node WHs
- **Grid/RandomizedSearchCV**.n_jobs = -1 → in parallel (distributed), using all processors
- joblib.parallel_backend → change def backend used by Parallel inside a with block
- disable_distributed_hpo → disable def optim + n_jobs=1 to minimize concurrency
- for multi-node WH → estimator fits are executed in parallel across all available cores on all nodes

MODEL VALIDATION AND EVALUATION



ML PIPELINES WITH SNOWPARK ML (IN CORTEX)

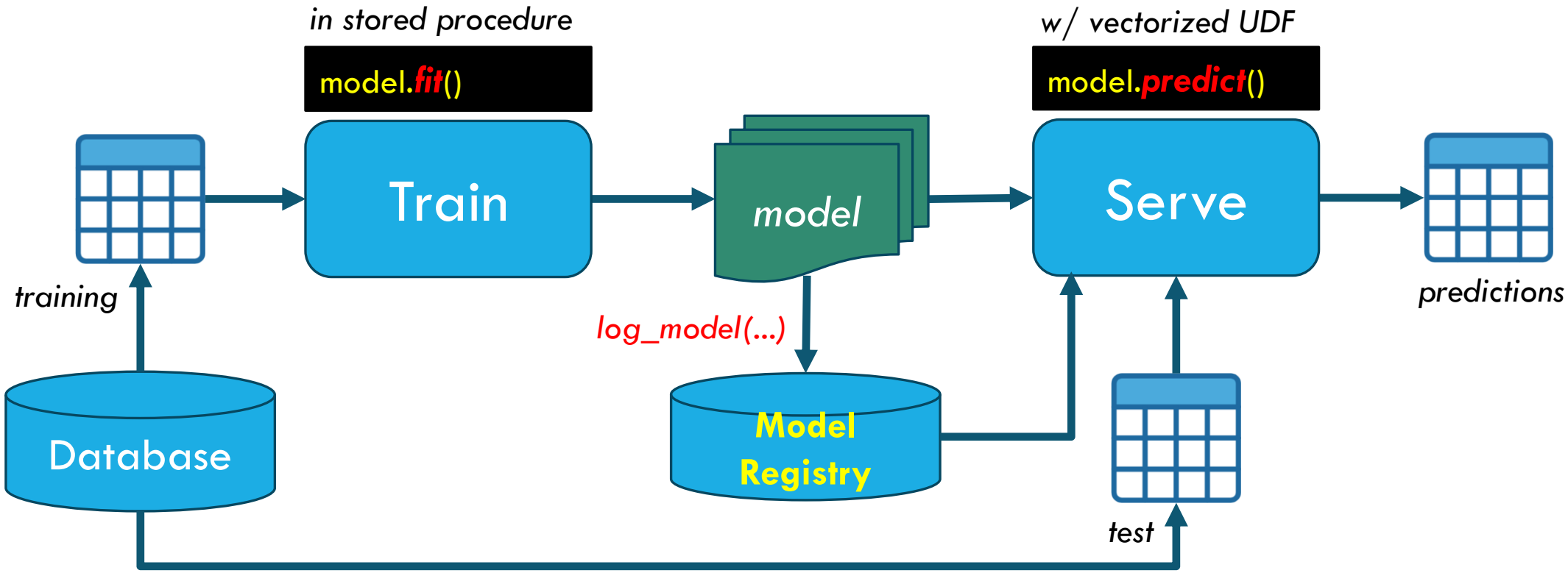




DISTRIBUTED METRICS (FOR MODEL EVALUATION)

- correlation, covariance
- accuracy/f1/fbeta/recall_score
- confusion_matrix
- log_loss
- precision_score/recall_fscore_support
- MAE, MAPE, MSE

SNOWPARK MLOPS ARCHITECTURE



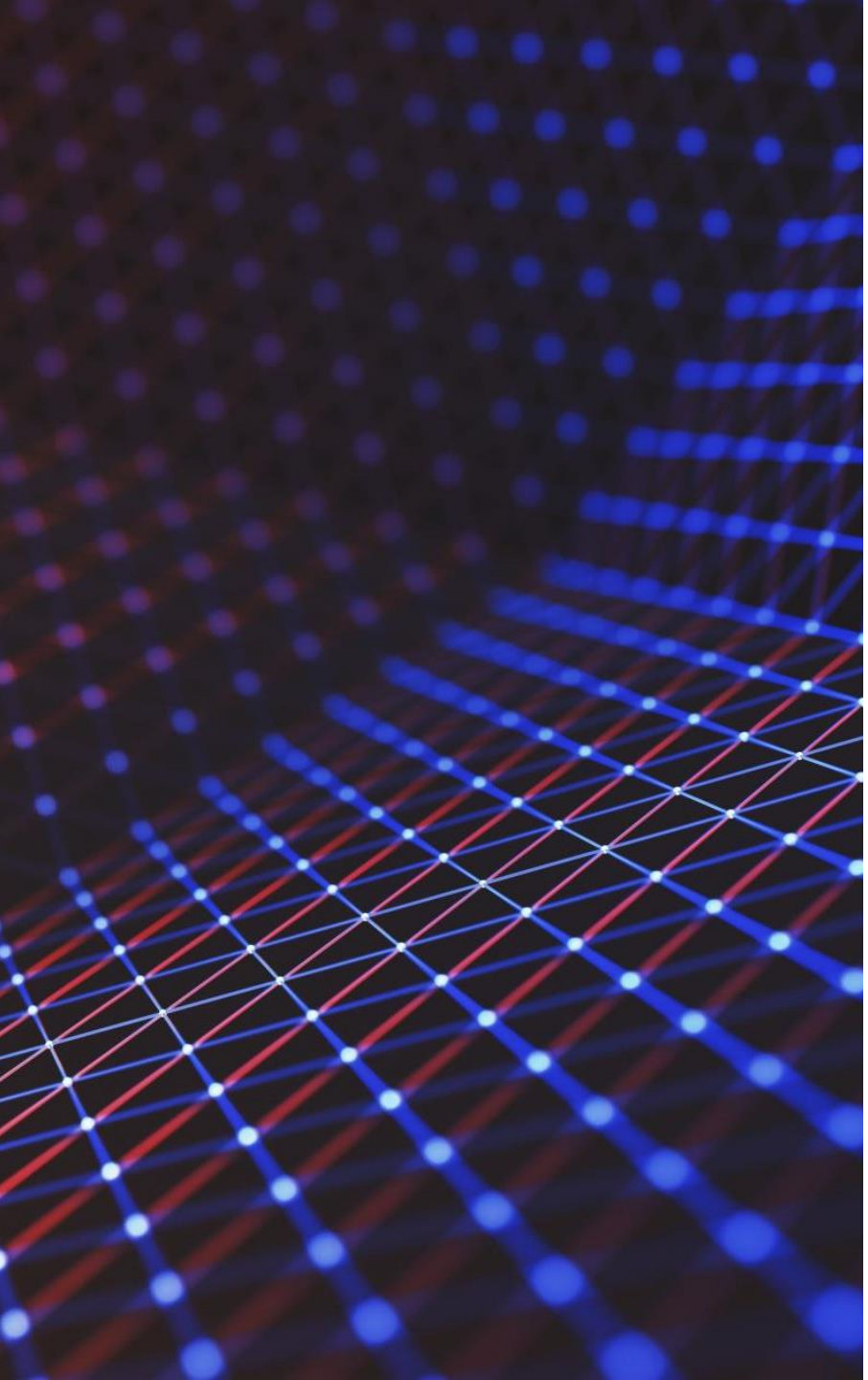
MLOps = ML + DEV + OPS

Develop

- Algorithm Training + Testing
- ETL (Data Pipelines)
- Continuous Integtation / Continuous Deployment

Operate

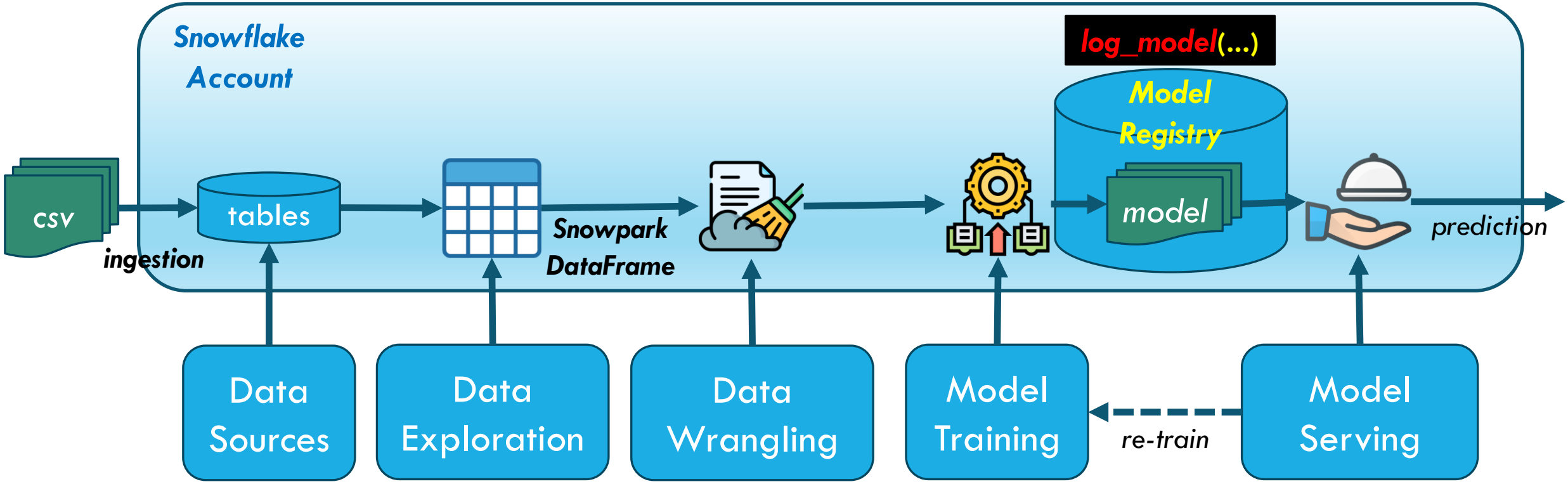
- Continuous Delivery
- Model Inference
- Monitoring and Management



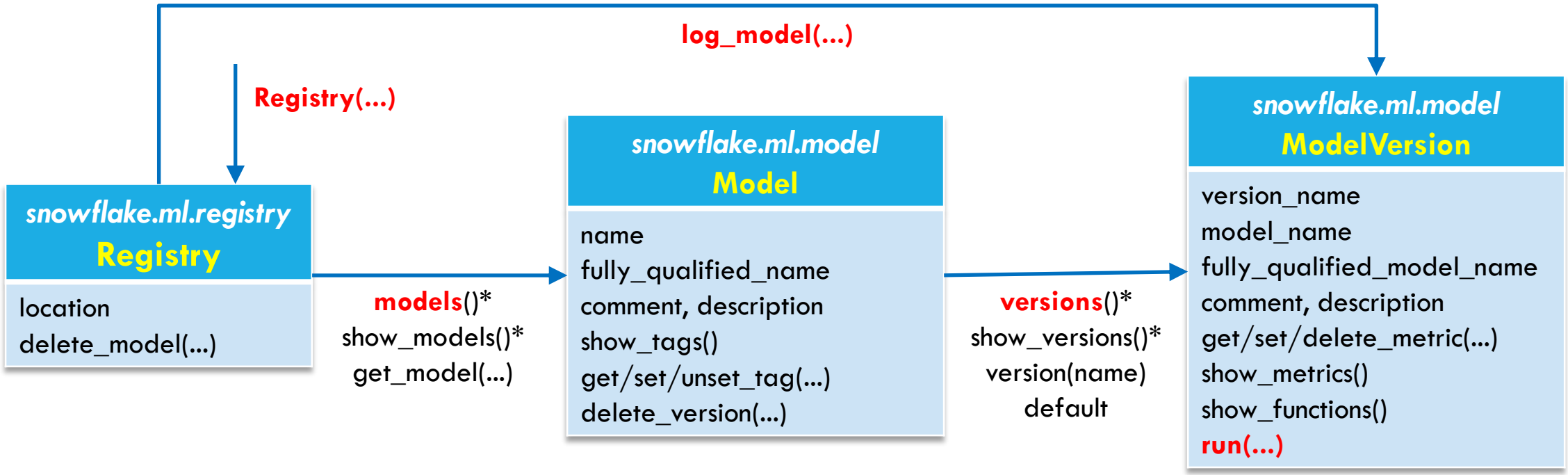
OPERATIONS (X-OPS)

- **MLOps (ML Operations)** = ML + DEV + OPS = deploy/maintain ML models in prod, through a model registry (w/ versioning)
- DevOps (Development Operations) = required by MLOps, w/ Code Artifact/Build/Pipeline, Step Functions
- ModelOps (ML Model Operations) = lifecycle management of ML models
- LLMOps (LLM Operations) = lifecycle management of LLMs

ML PIPELINES WITH SNOWPARK ML (IN CORTEX)



MODEL REGISTRY API: OBJECT MODEL

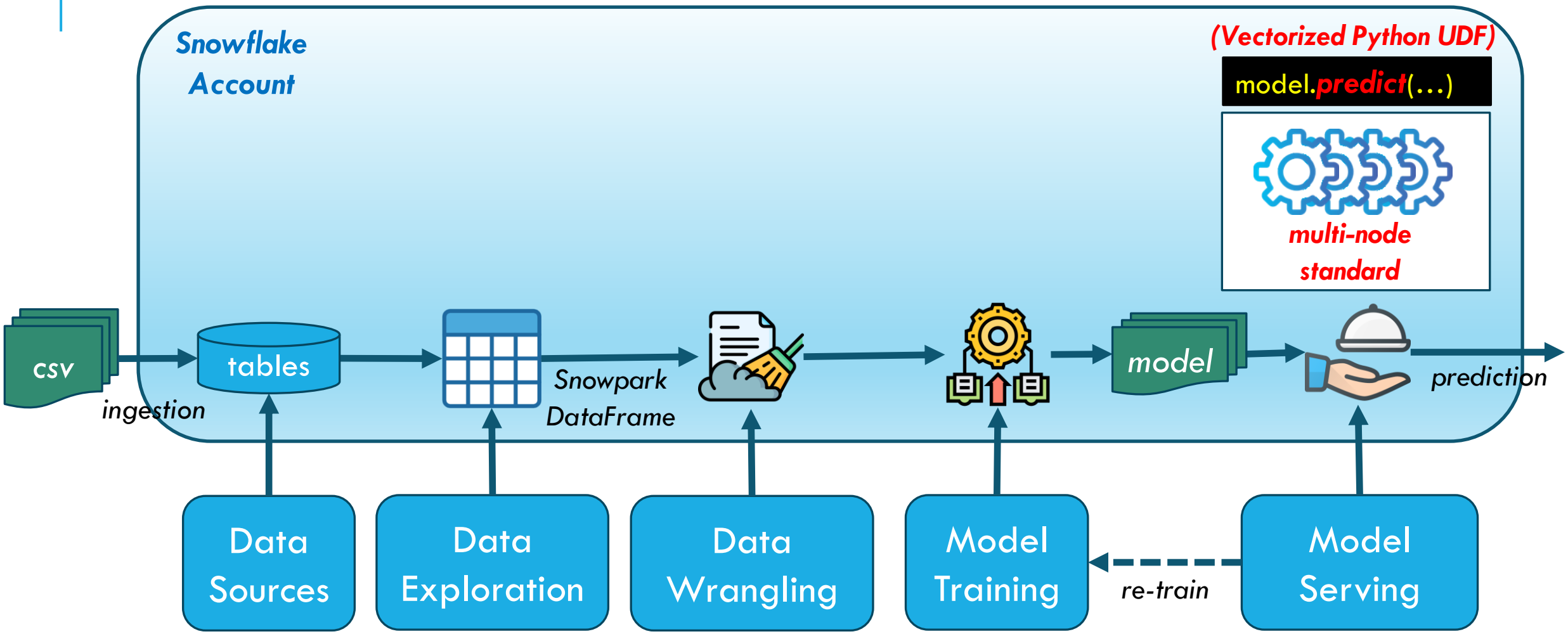


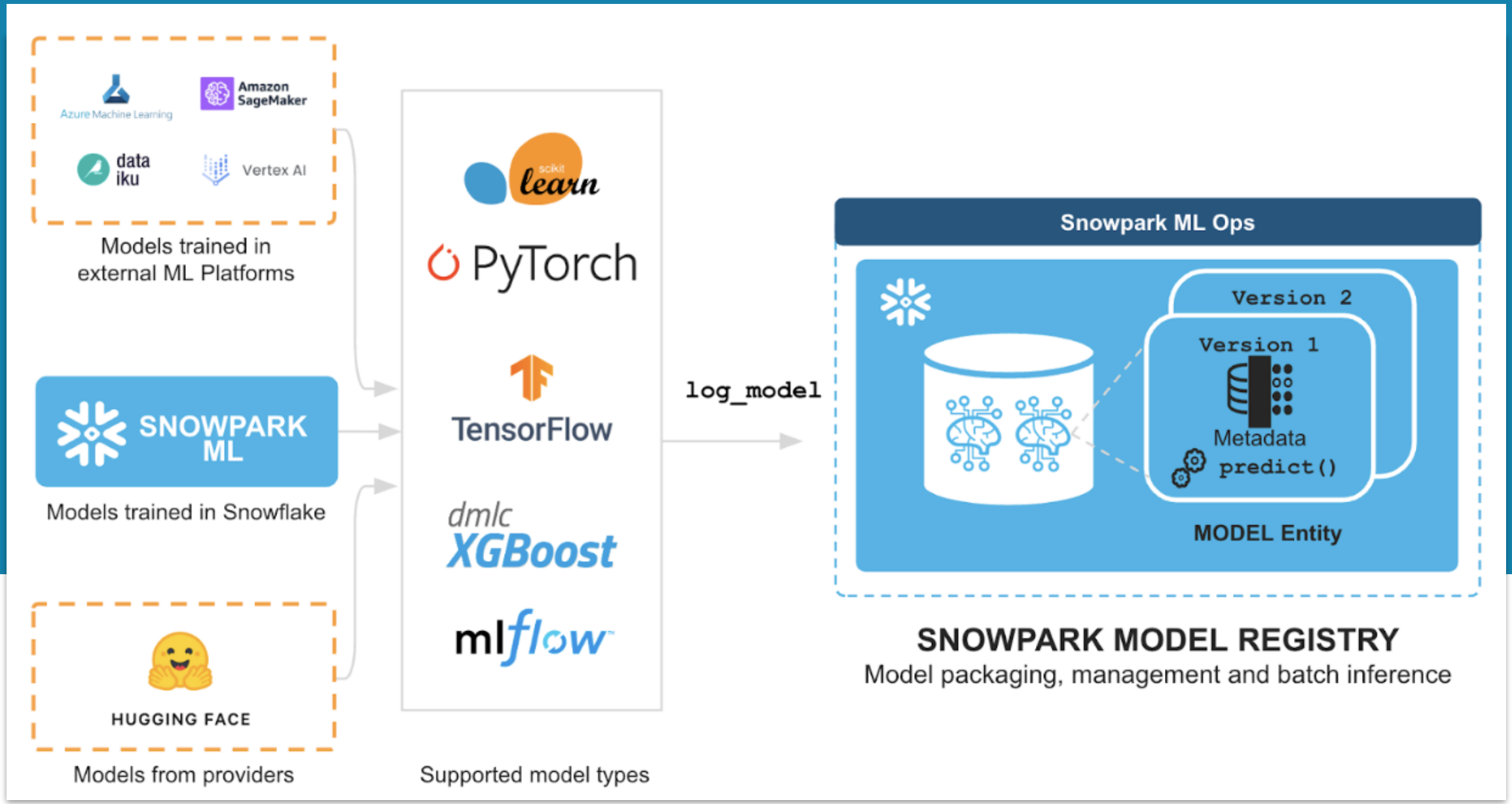
CREATE MODEL ...
SHOW MODELS
[IN DATABASE/SCHEMA ...]

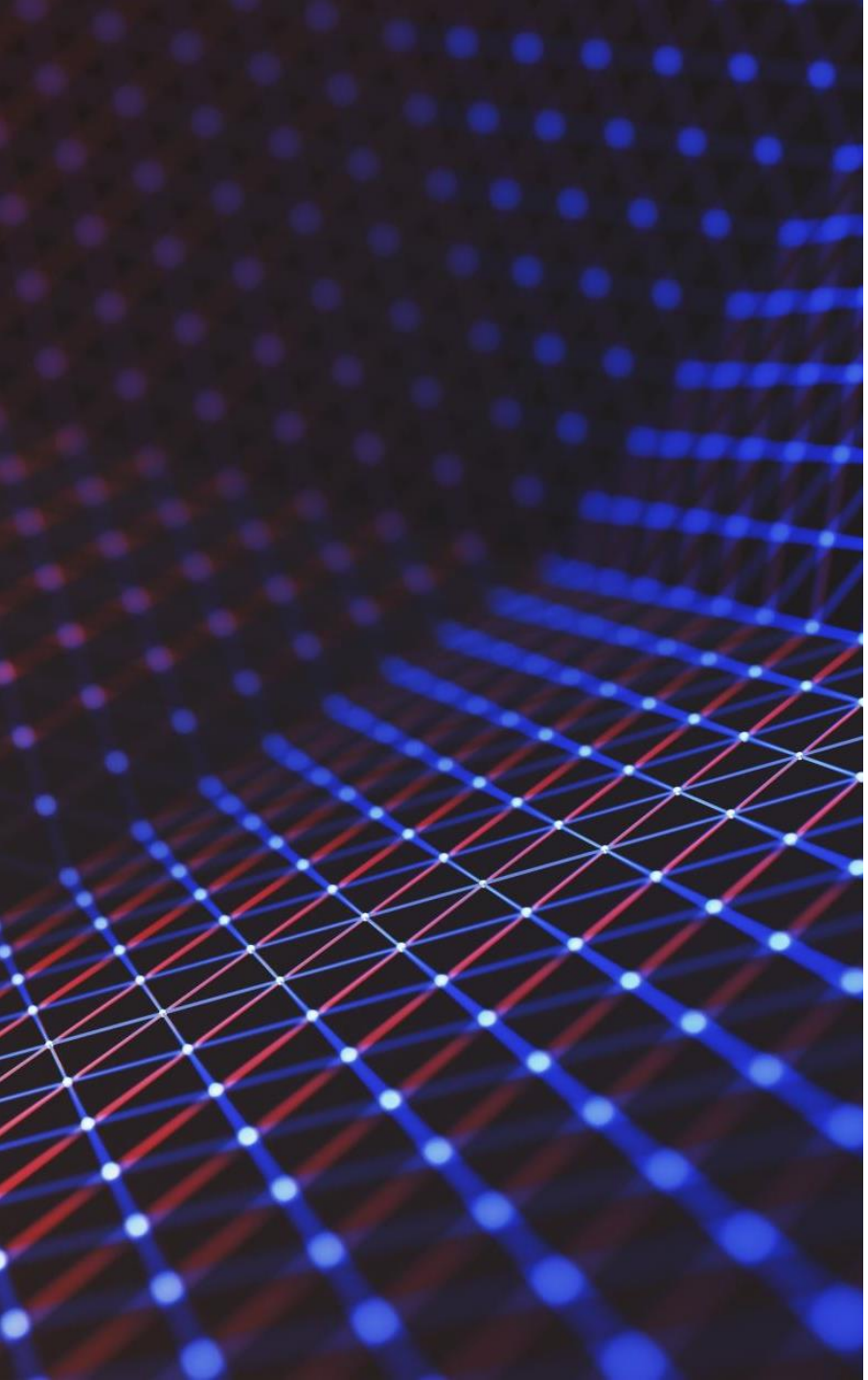
ALTER/DROP MODEL ...
SHOW VERSIONS [IN MODEL ...]

ALTER MODEL ...
ADD/MODIFY VERSION ...

ML PIPELINES WITH SNOWPARK ML (IN CORTEX)



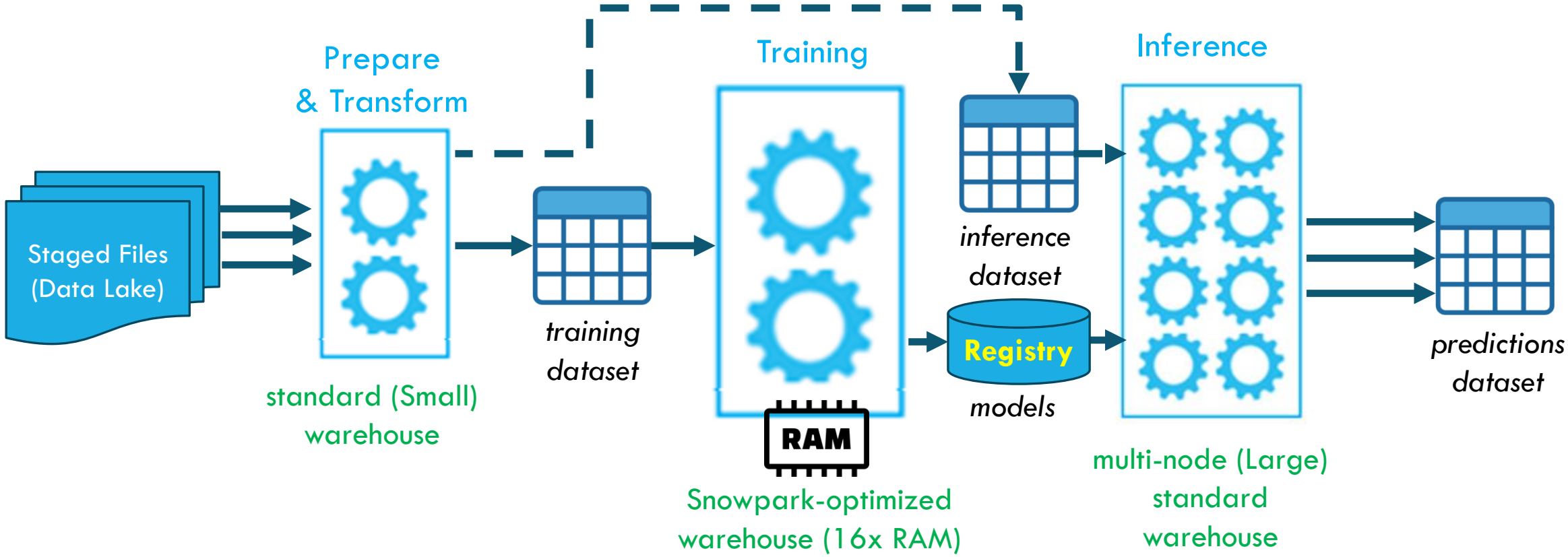

Azure Machine Learning
Amazon SageMaker
data iku
Vertex AI
Models trained in external ML Platforms
SNOWPARK ML
Models trained in Snowflake
HUGGING FACE
Models from providers
scikit learn
PyTorch
TensorFlow
dmlc XGBoost
mlflow
Supported model types
log_model
Snowpark ML Ops
Version 2
Version 1
Metadata
predict()
MODEL Entity
SNOWPARK MODEL REGISTRY
Model packaging, management and batch inference



MODEL TYPES & PROVIDERS

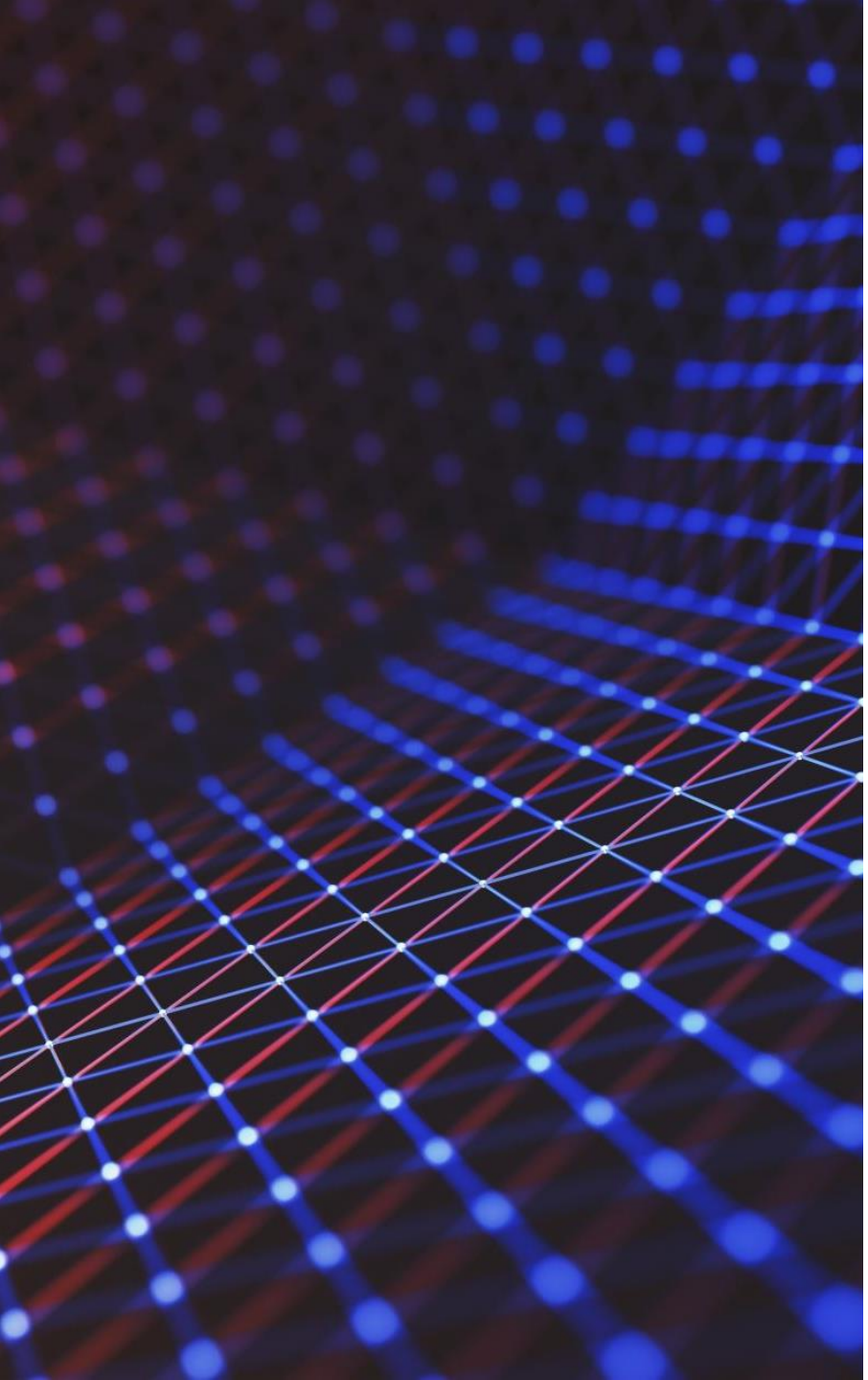
- *model types*
  - scikit-learn
  - XGBoost
  - PyTorch
  - TensorFlow
  - MLFlow PyFunc
- *model providers*
  - **Snowpark ML Modeling**
  - HuggingFace pipeline
  - external ML platforms

SELECTION OF WAREHOUSES



VIRTUAL WAREHOUSE (COMPUTE) CREDITS

Type of Virtual Warehouse	X-Small	Small	Medium	Large	X-Large	2X-Large	3X-Large	4X-Large	5X-Large	6X-Large
Standard	1	2	4	8	16	32	64	128	256	512
Snowpark-optimized	n/a	n/a	6	12	24	48	96	192	384	768

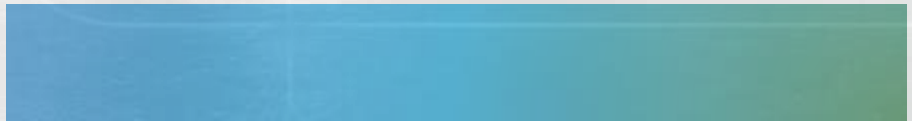


MODEL REGISTRY COSTS

- ***storage*** → for model artifacts/metadata/functions
- ***data transfer*** → copy files between stages to Snowflake
- ***serverless compute*** → show models/versions + alter comments/tags/metrics.
- ***warehouse compute*** → for model/version creation + invoke model's methods

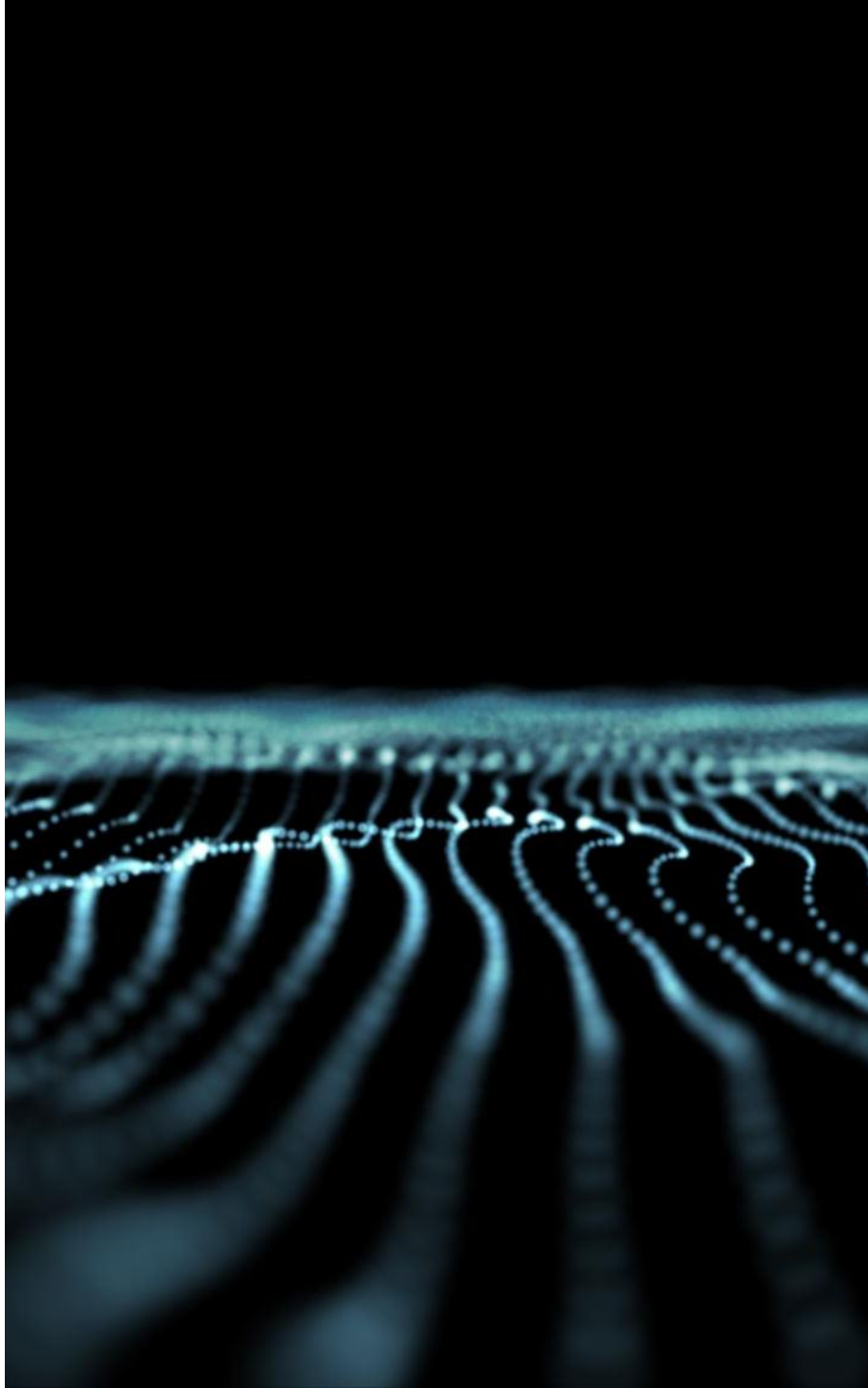
ML-POWERED FUNCTIONS AND CLASSES

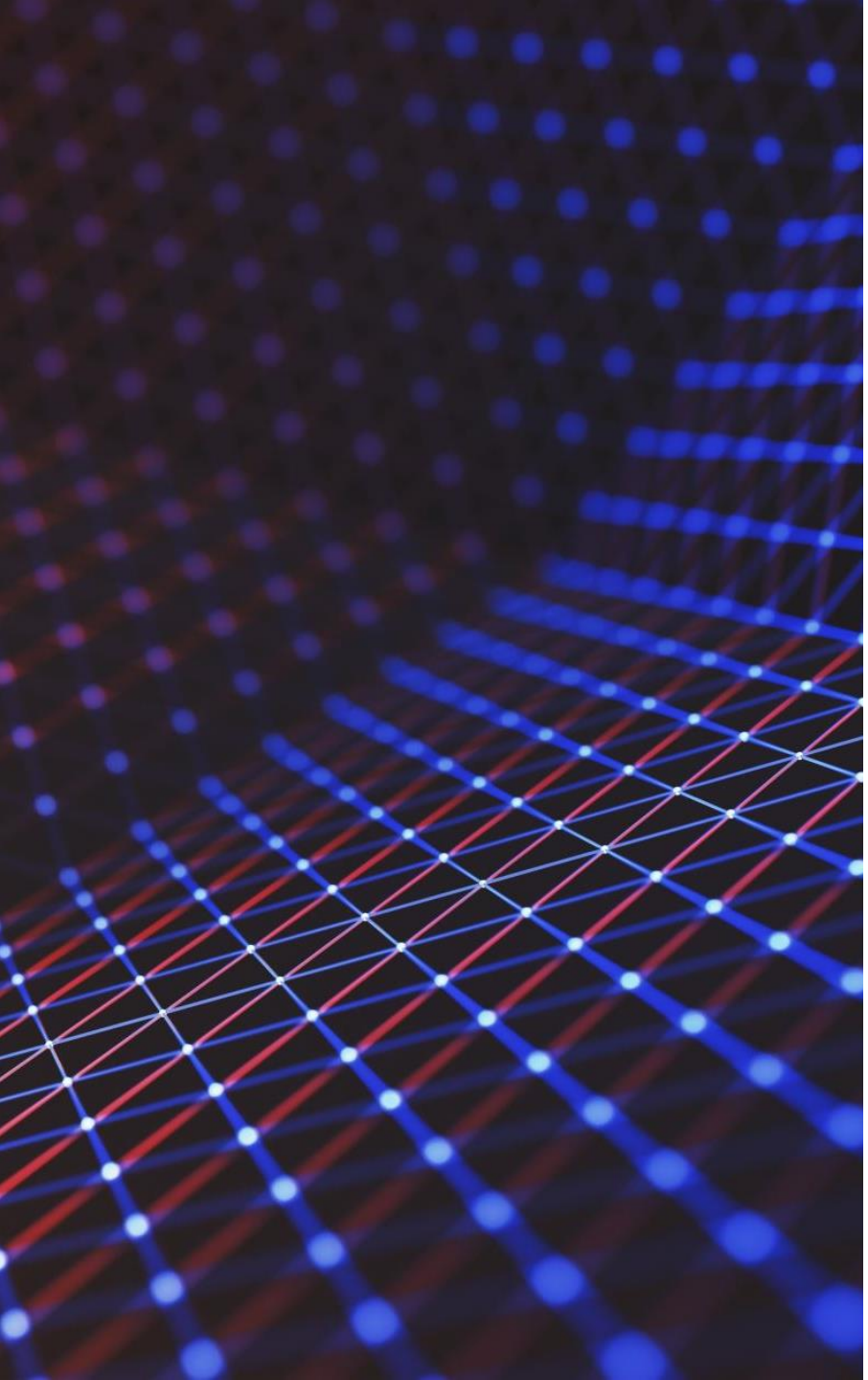
- Time-Series Forecasting
- Anomaly Detection
- Classification
- Contribution Explorer



ALL ML-POWERED FUNCTIONS AND CLASSES

- **Classification** = extract sorted categories from features
  - SNOWFLAKE.ML.**CLASSIFICATION** class
  - model!***PREDICT*** class instance method
- **Time-Series Forecasting** = predicts next N future values for single/multiple TS
  - SNOWFLAKE.ML.**FORECAST** class
  - model!***FORECAST*** class instance method
- **Anomaly Detection** = finds potential outliers in a TS by comparing data to a forecast of the same time period
  - SNOWFLAKE.ML.**ANOMALY_DETECTION** class
  - model!***DETECT_ANOMALIES*** class instance method
- **Contribution Explorer** = dets factors influencing a metric
  - SNOWFLAKE.ML.***TOP_INSIGHTS***(...) secure Python UDTF



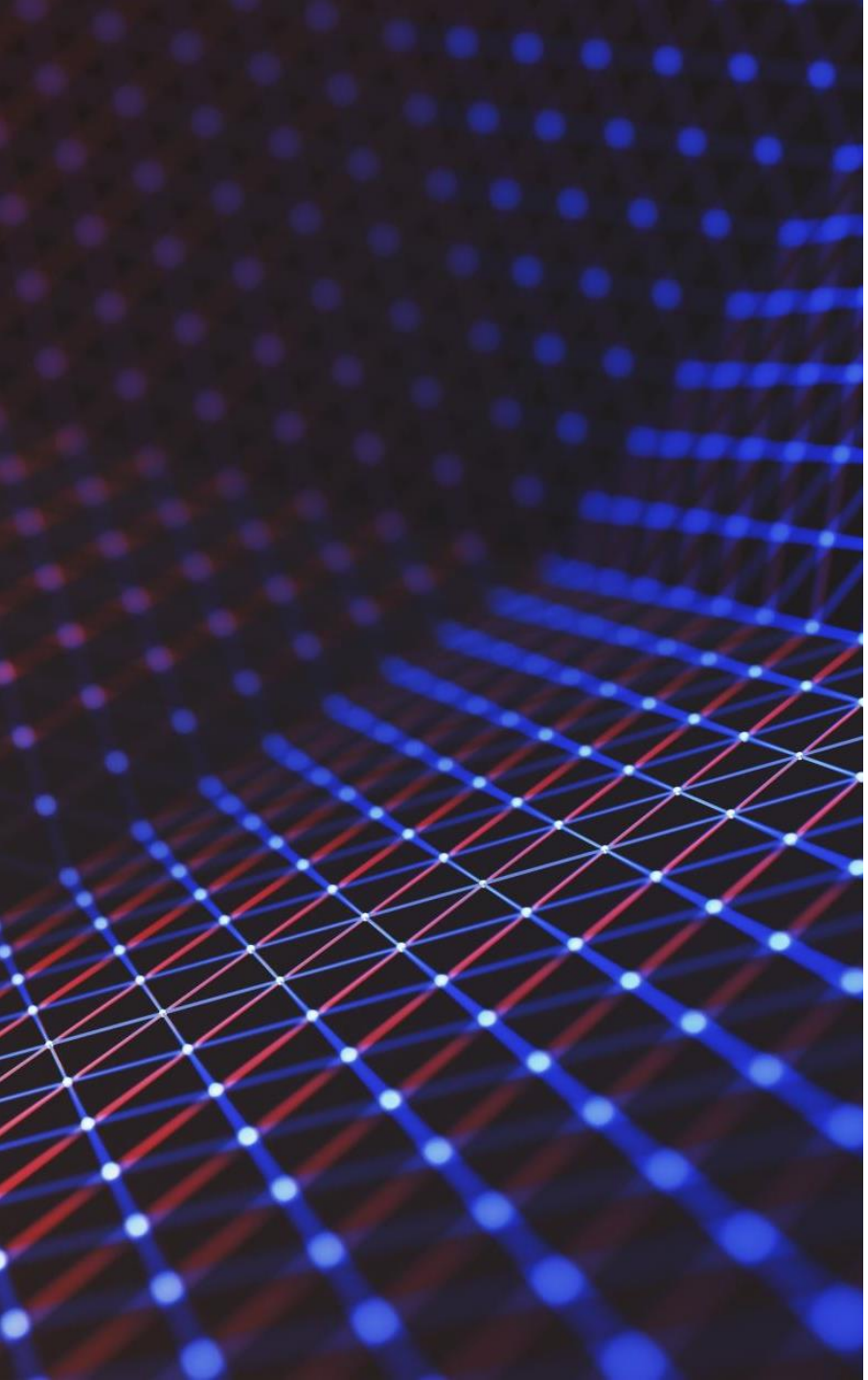


ALL BUILTIN SNOWFLAKE CLASSES

- ***SNOWFLAKE.ML*** *schema*
  - **CLASSIFICATION** = sorts data into categories based on features in the data
  - **FORECAST** = model that produces a forecast for single/multiple TSs
  - **ANOMALY_DETECTION** = detect outliers in your TS data

- ***SNOWFLAKE.CORE.*****BUDGET** = monitor credit usage of supported objects

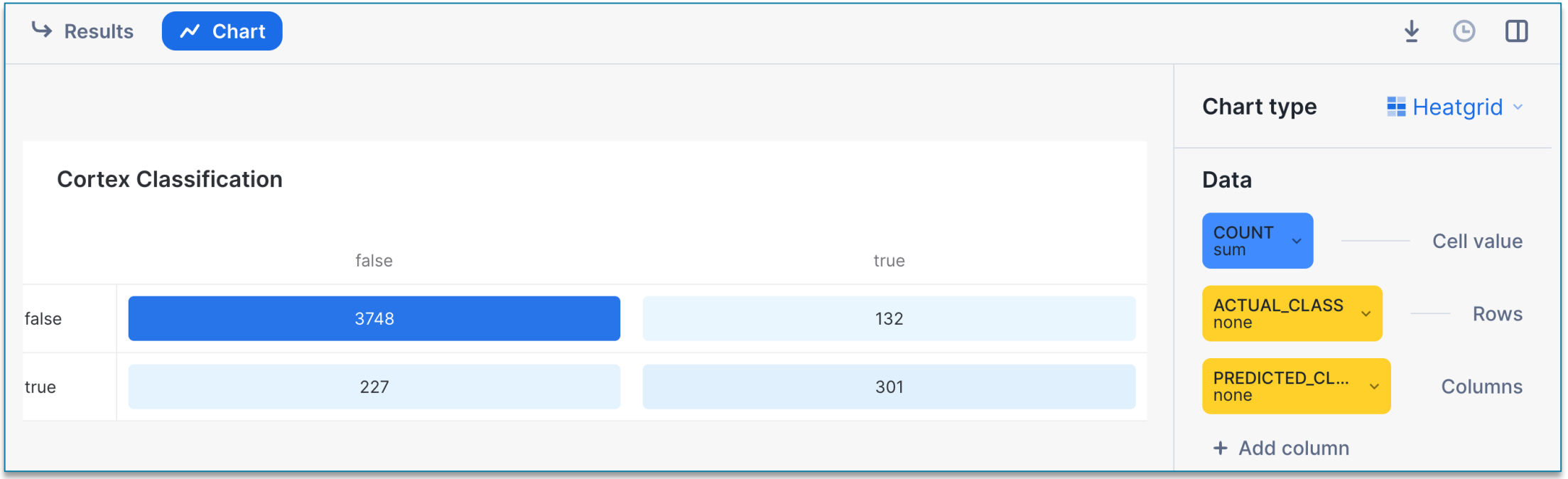
- ***SNOWFLAKE.DATA_PRIVACY.*****CUSTOM_CLASSIFIER** = to extend your data classification capabilities

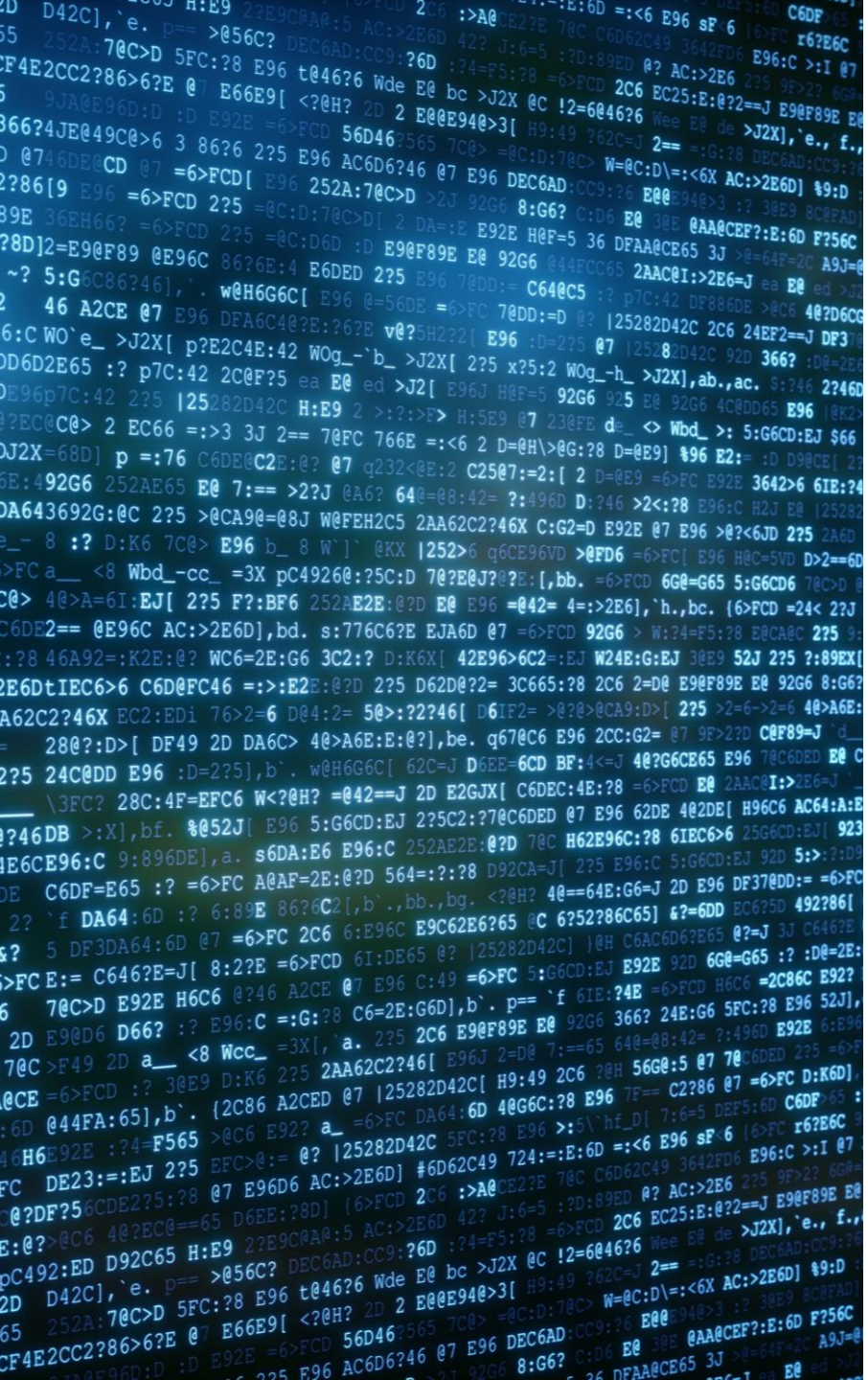


SQL SNOWFLAKE CLASSES

- **class** = SQL object, at the schema level - built-in only (for now)
- w/ stored procs + functions → *class methods*, to be called from instances
- private state + private procedures/functions
- CREATE class_name *instance_name*(...) → named instance obj created from a class
- *instance_name*!class_method(...) - call method on instance
- no static methods, but could have built-in predefined instances → see *account_root_budget* singleton instance
- modify *search path* → to include schema ML, for session/user/account
- SQL commands: CREATE / ALTER / SHOW / DROP

CLASSIFICATION (CONFUSION MATRIX)

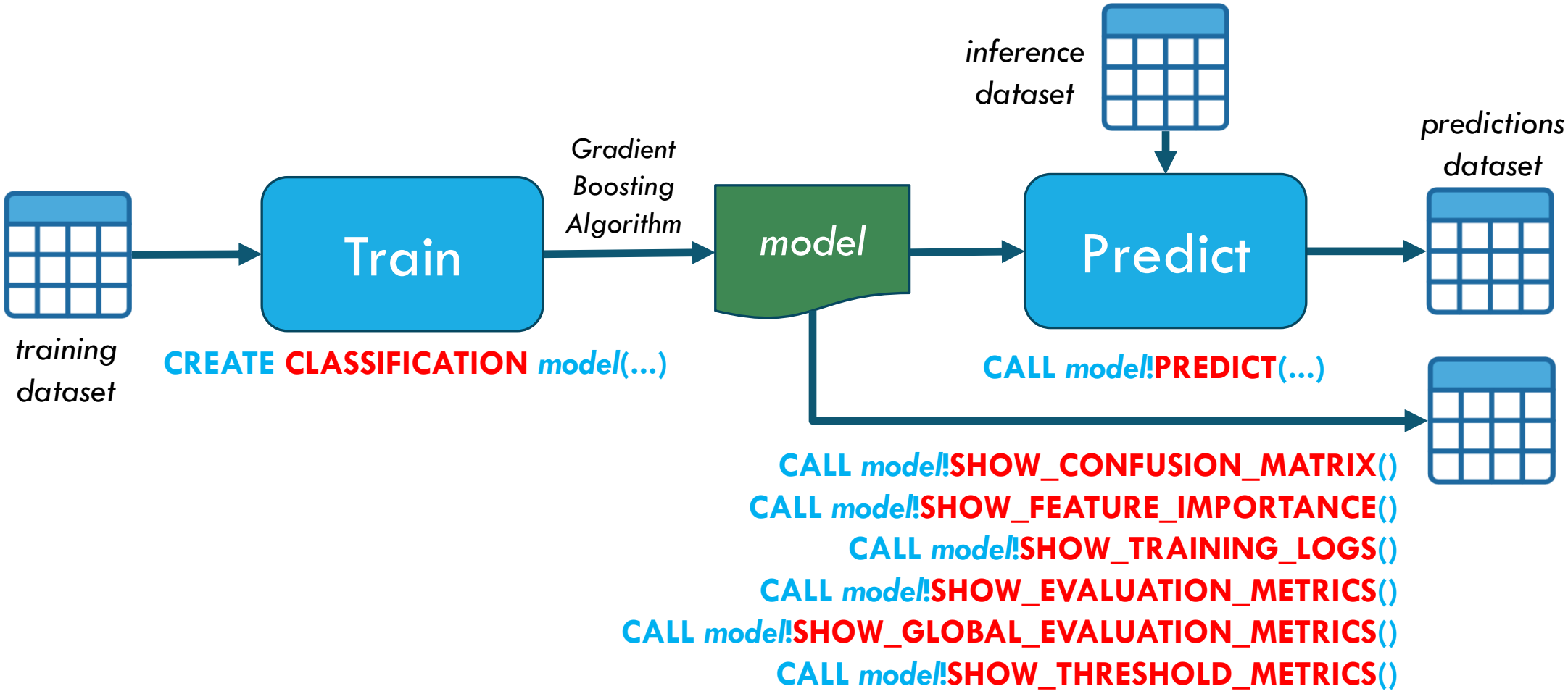




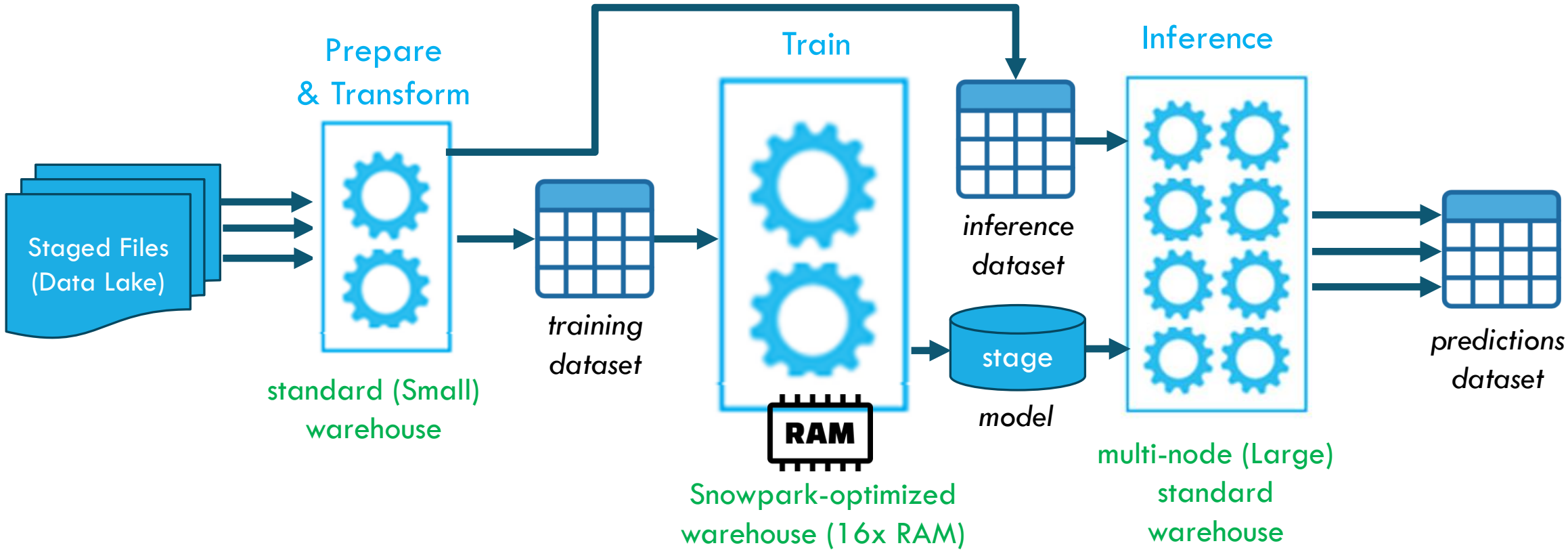
CLASSIFICATION

- = for binary/multi-class classification
- ex: for churn prediction / credit card fraud detection / spam detection (True/False)...
- w/ **GBM (Gradient Boosting Machine)** alg
  - *binary* → model trained w/ ***AUC*** loss function
  - *multi-class* → ***logistic*** loss function
- *training datasets*
  - target column = labeled class of each data point
  - 1+ feature columns
- for continuous (numeric, cast to string) / categorical (boolean/string)

CLASSIFICATION ARCHITECTURE

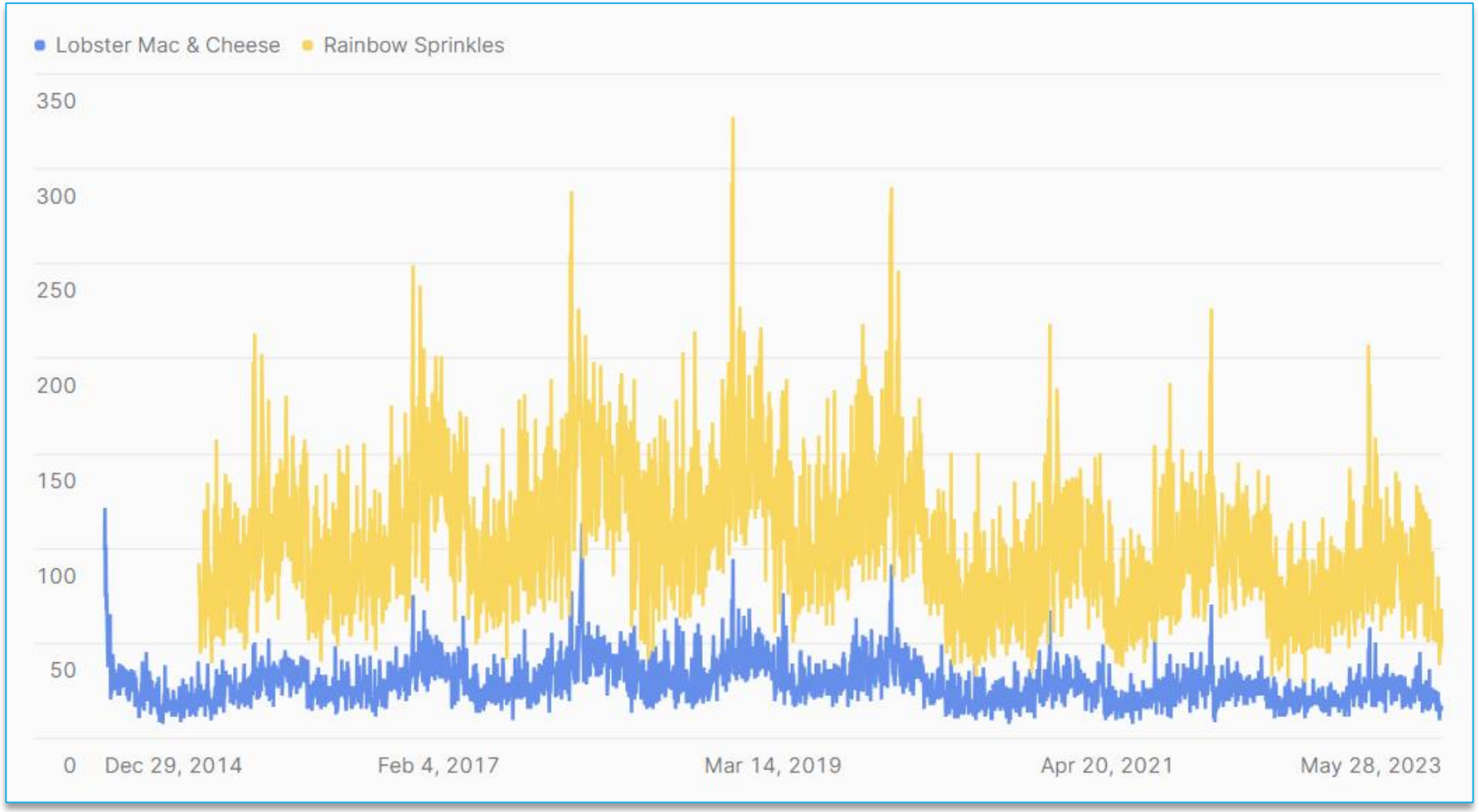


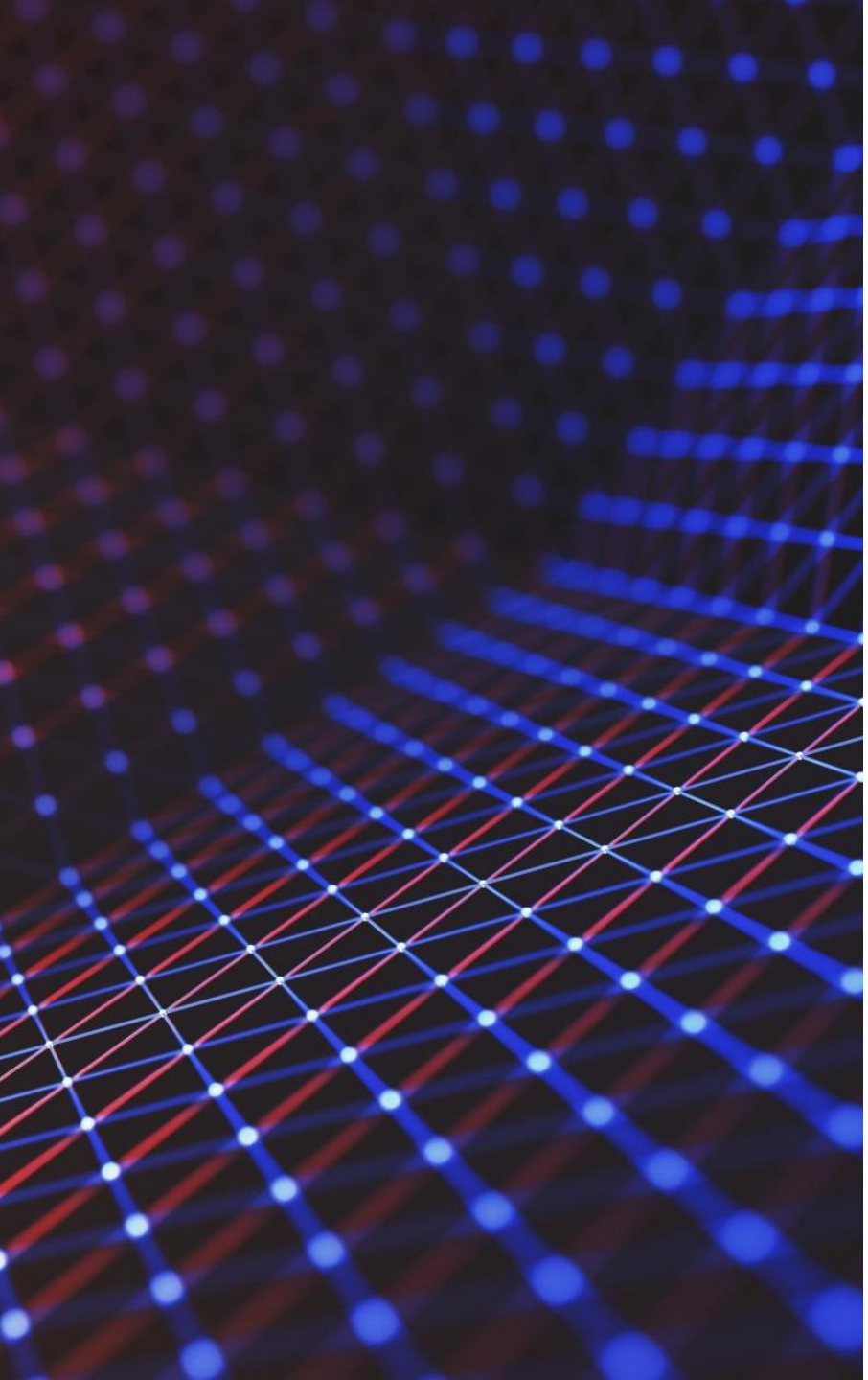
CLASSIFICATION PIPELINE



TIME SERIES FORECASTING

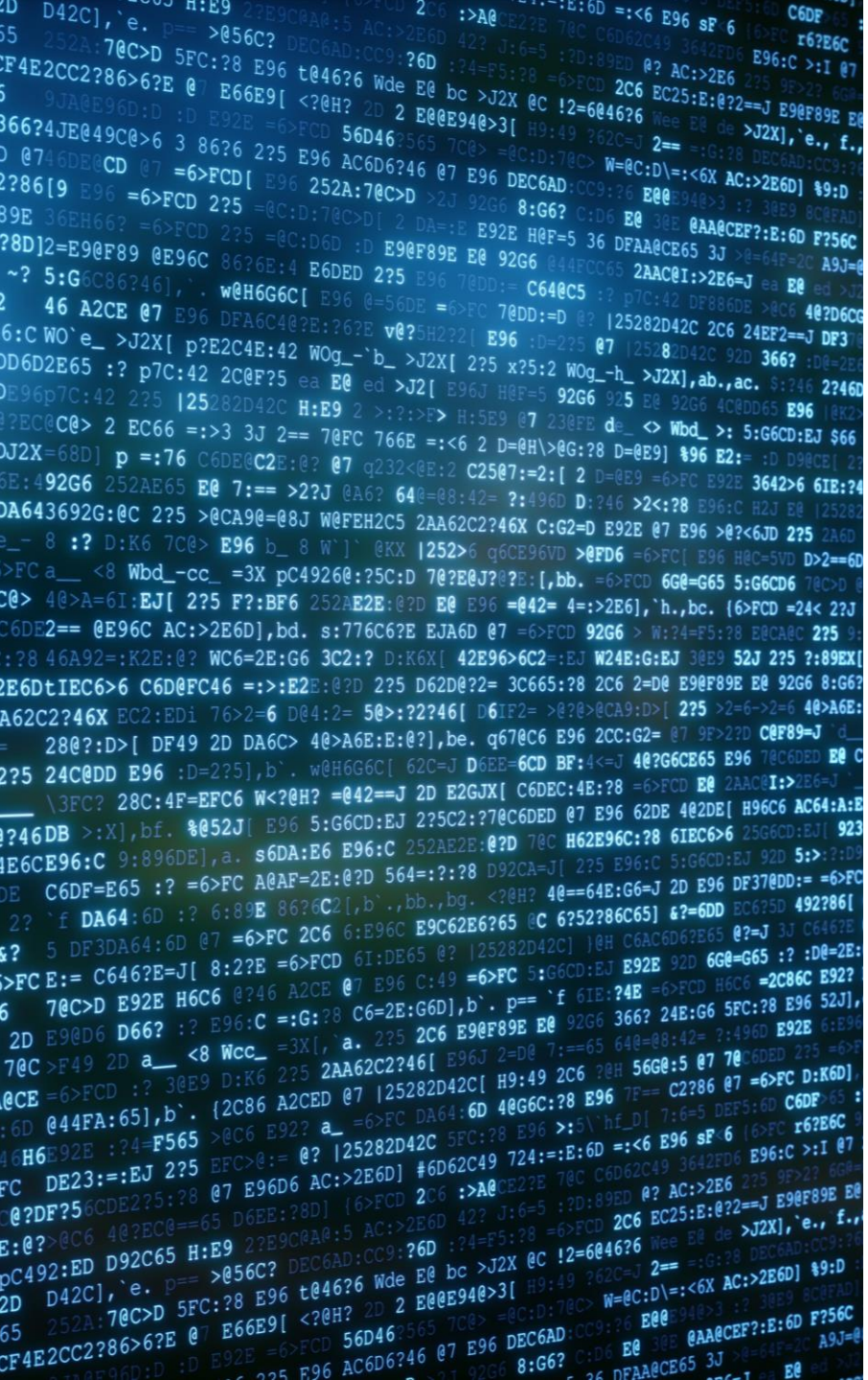
MULTI-SERIES HISTORICAL DATA





TRAINING DATASET (HISTORICAL/INPUT)

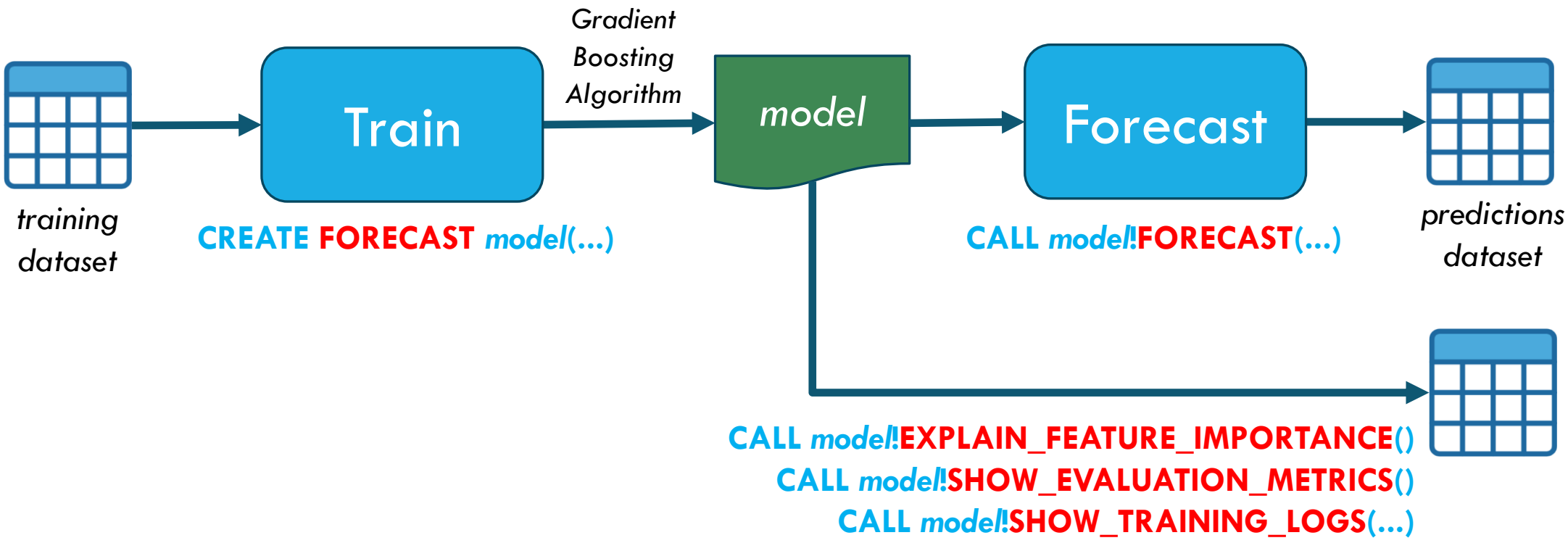
- **timestamp** = w/ fixed regular freq (secs/hourly/every 5 mins/...) – use DATE_TRUNC/TIME_SLICE if irregular
- **target** = quantity of interest, at each timestamp (min 12 values)
- **exogenous vars** = independent vars (**endogenous** for dependent/correlated vars) → optional, num/categ
  - weather data: temperature, rainfall...
  - company-specific info: historic/planned holidays, advertisement campaigns, event schedules...
- **single/multi-series** = ex: each store sales can be forecasted separatelly, w/ store id



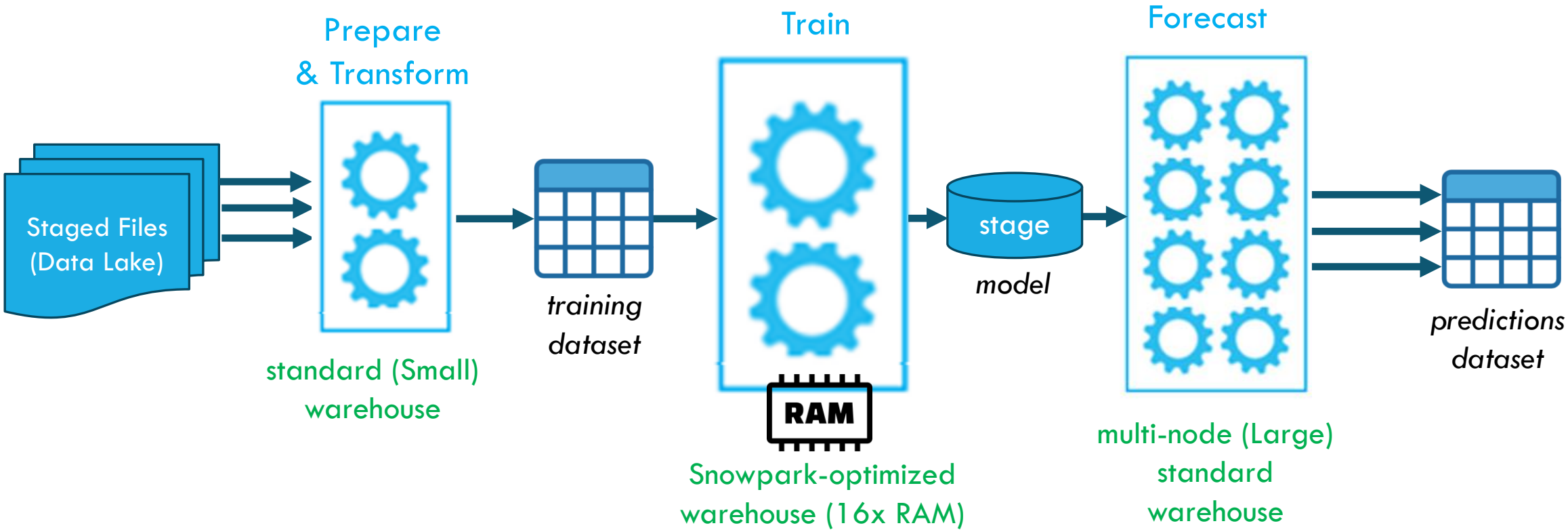
FORECAST DATASET (PREDICTION/OUTPUT)

- **forecast(s)** = predicted future values based on factors that influence most data, from model trained on historical data (on past values)
- **trend/seasonality/seasonal amplitudes** = internally inferred from the data (on min 1 minute seasonality)
- **prediction intervals** = estimated range in which a certain percentage of data is likely to fall (95% by def)

MODEL TRAINING & FORECASTING ARCHITECTURE



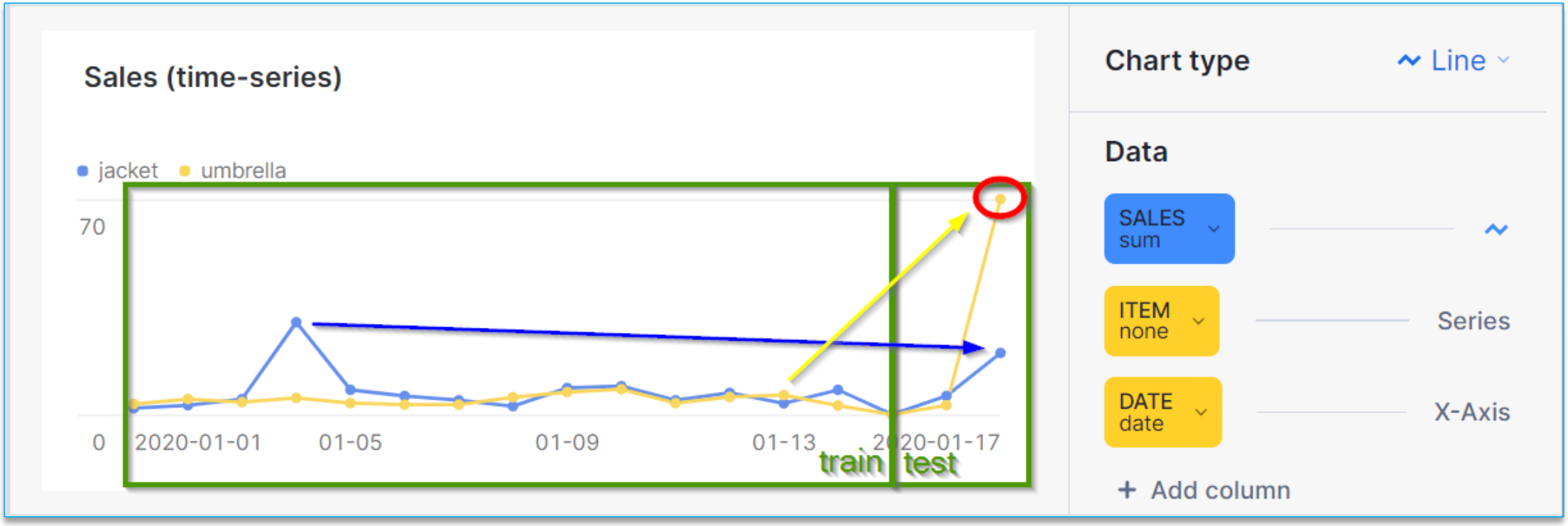
MODEL TRAINING & FORECASTING PIPELINE

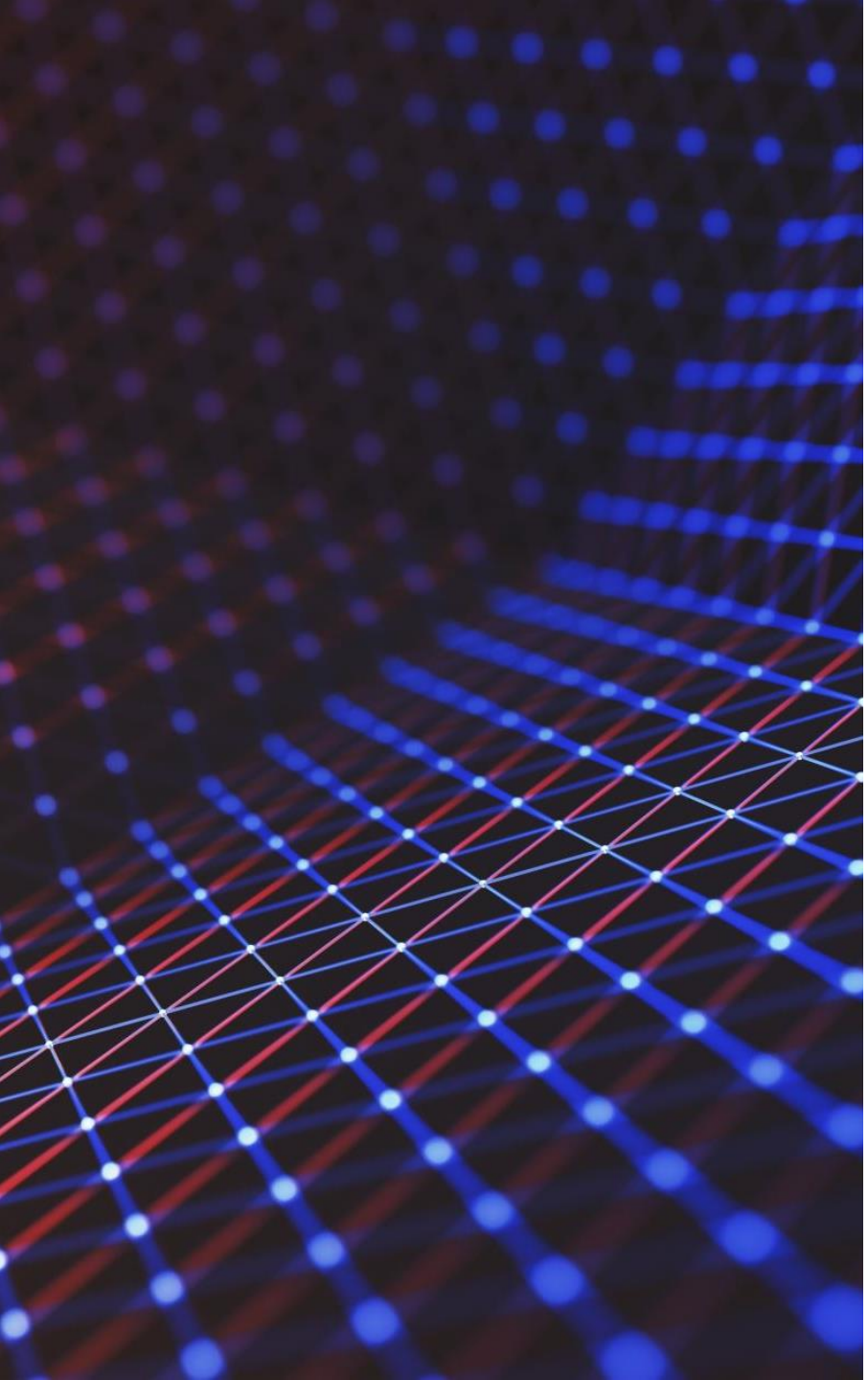


ANOMALY DETECTION ALGORITHM

The algorithm compares time-series test data to a forecast of the same time period from the training data

ANOMALY DETECTION WITH MULTIPLE TIME-SERIES

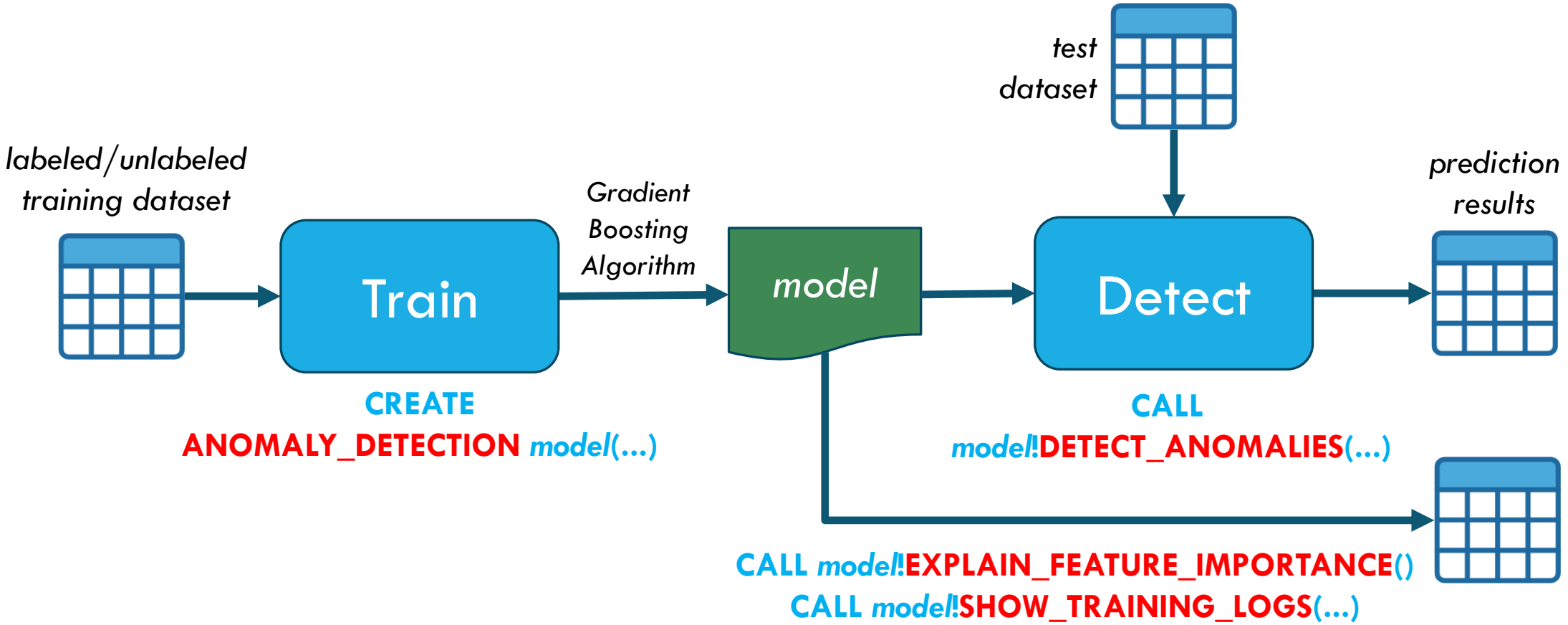




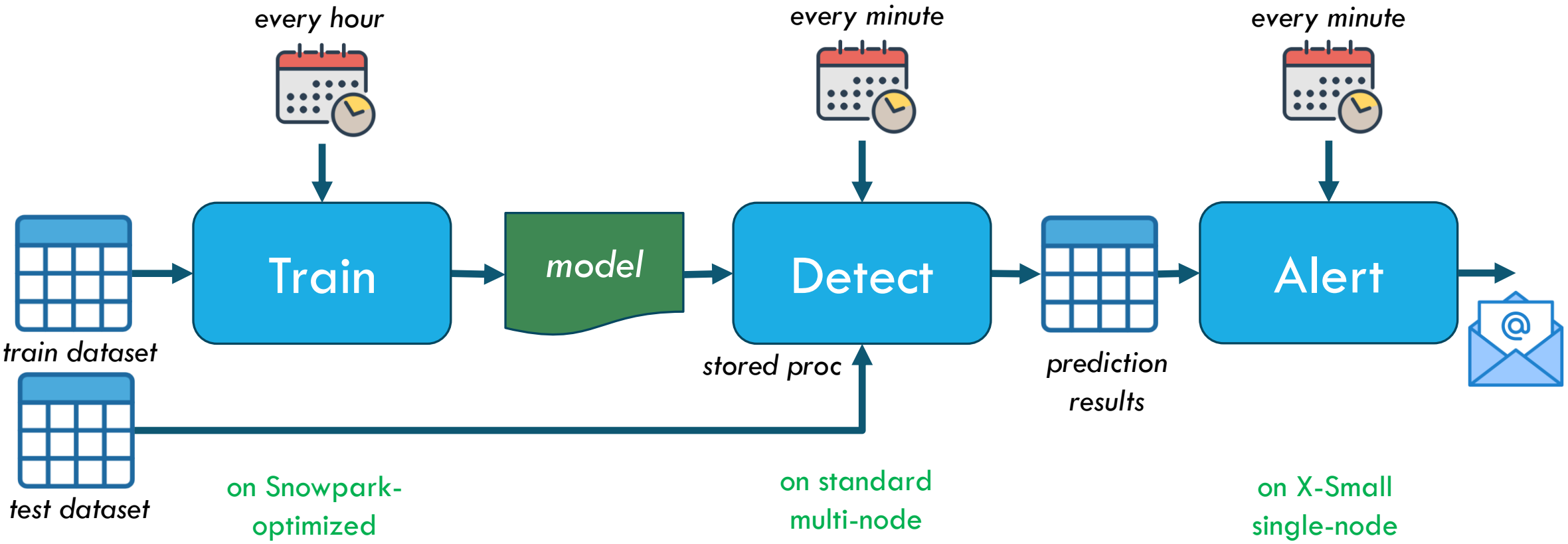
ANOMALY DETECTION

- = find potential **outliers** in TS data
- w/ same GBM algorithm as Forecasting/Classification
- *typical actions*
  - remove → to improve quality of results
  - monitor (w/ tasks) + alert when detected
- *datasets*
  - **train** = labeled/unlabeled, w/ timestamp + target numeric column (like sales)
  - **test** = always unlabeled, to detect anomalies
- **unsupervised** (no outliers)/**supervised** (labeled outliers)
- **single-TS** / **multi-TSs** (ARRAY of column values)

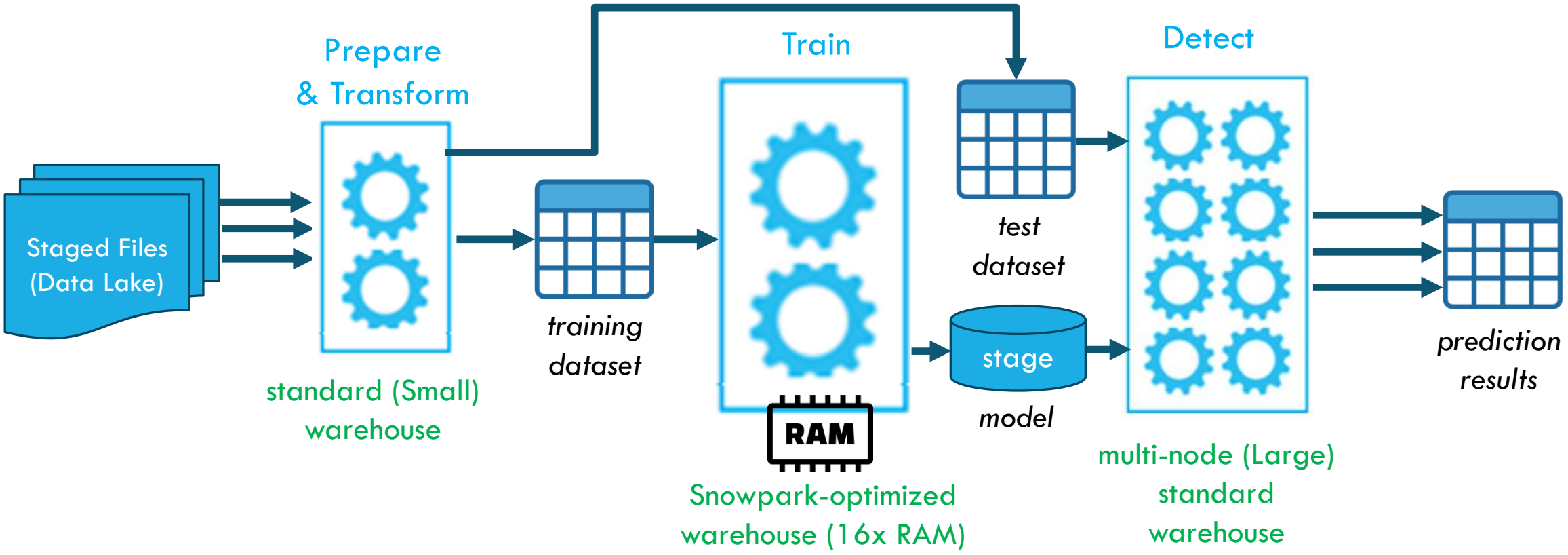
ANOMALY DETECTION ARCHITECTURE

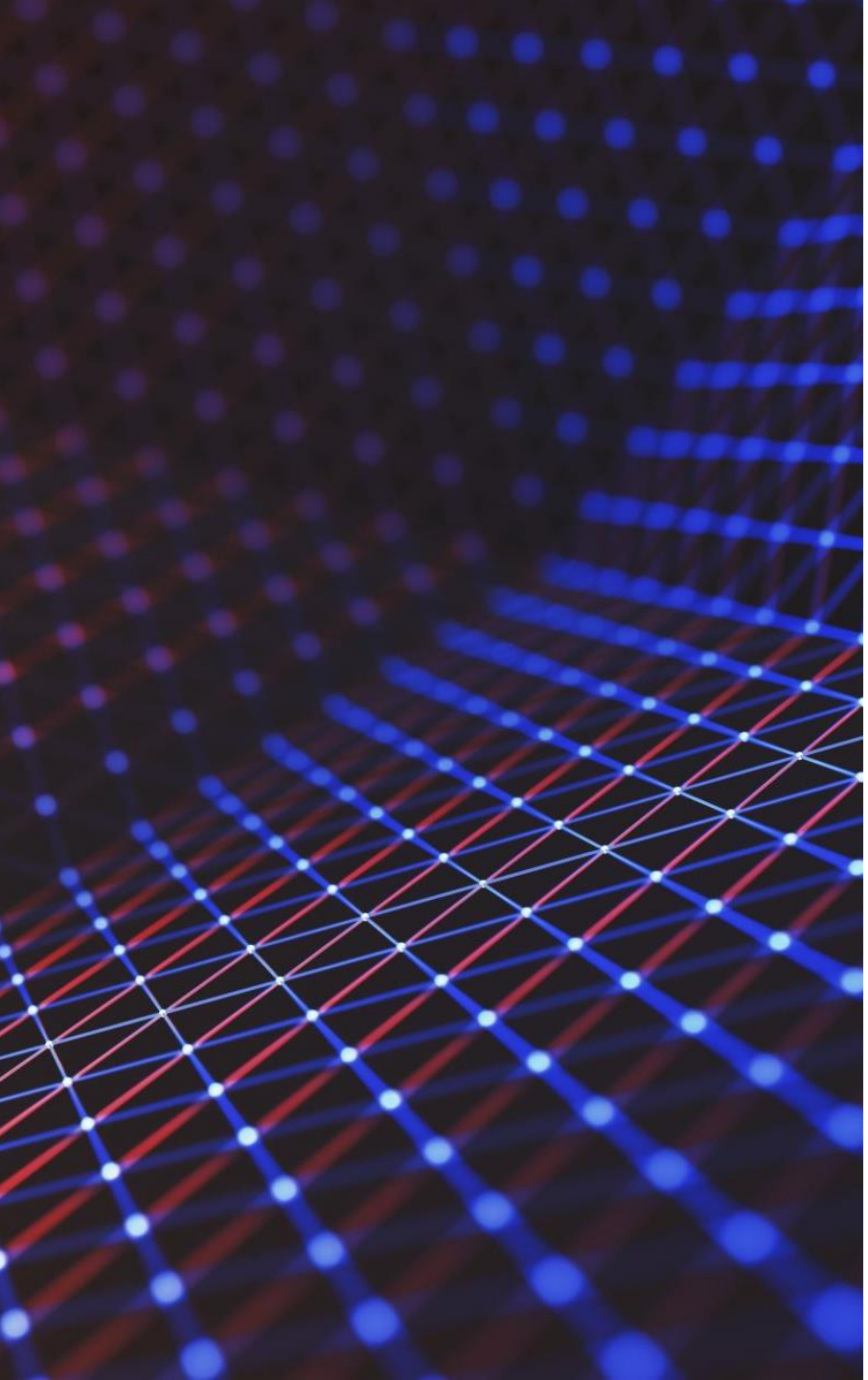


AUTOMATED ANOMALY DETECTION ARCHITECTURE



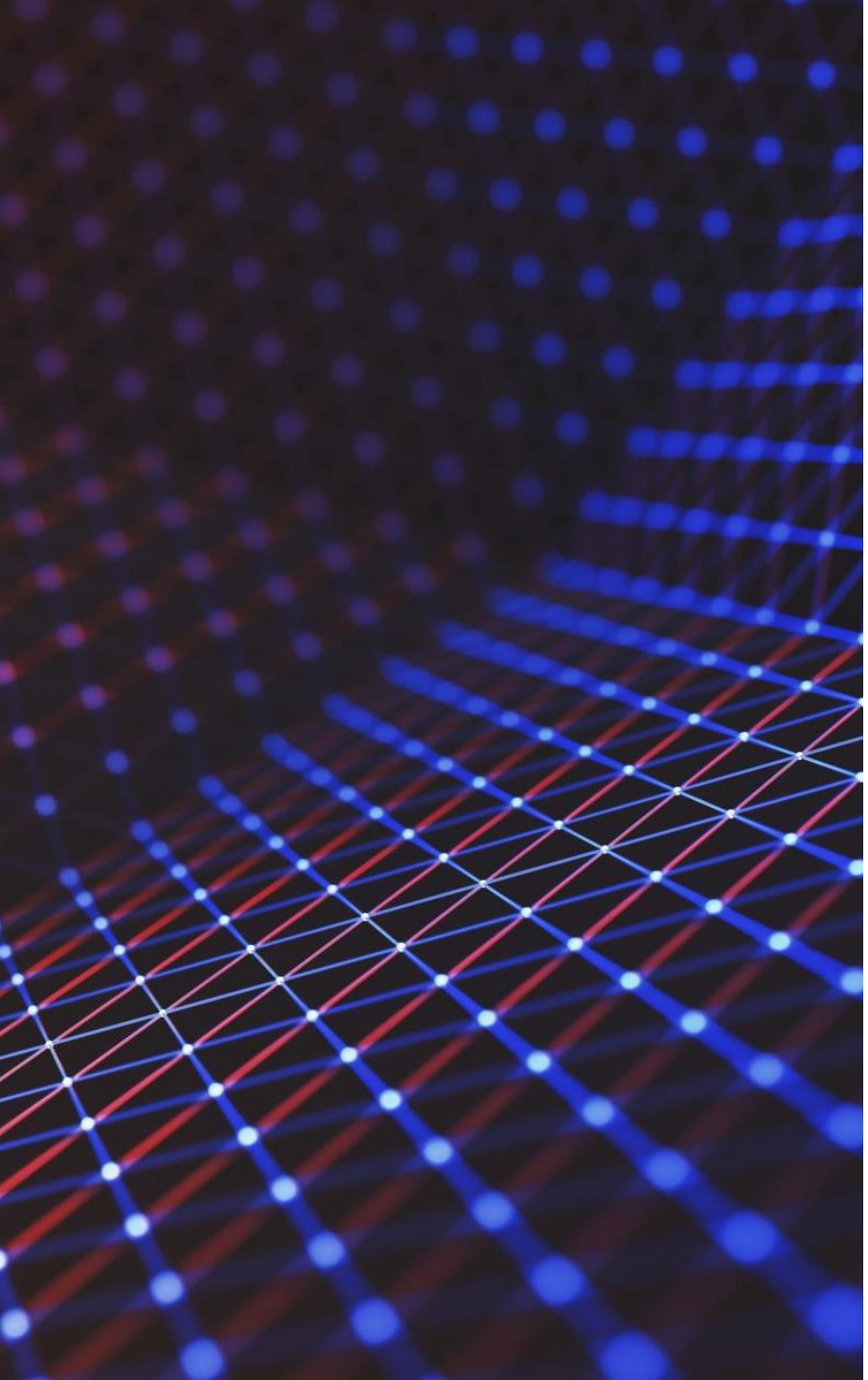
ANOMALY DETECTION PIPELINE





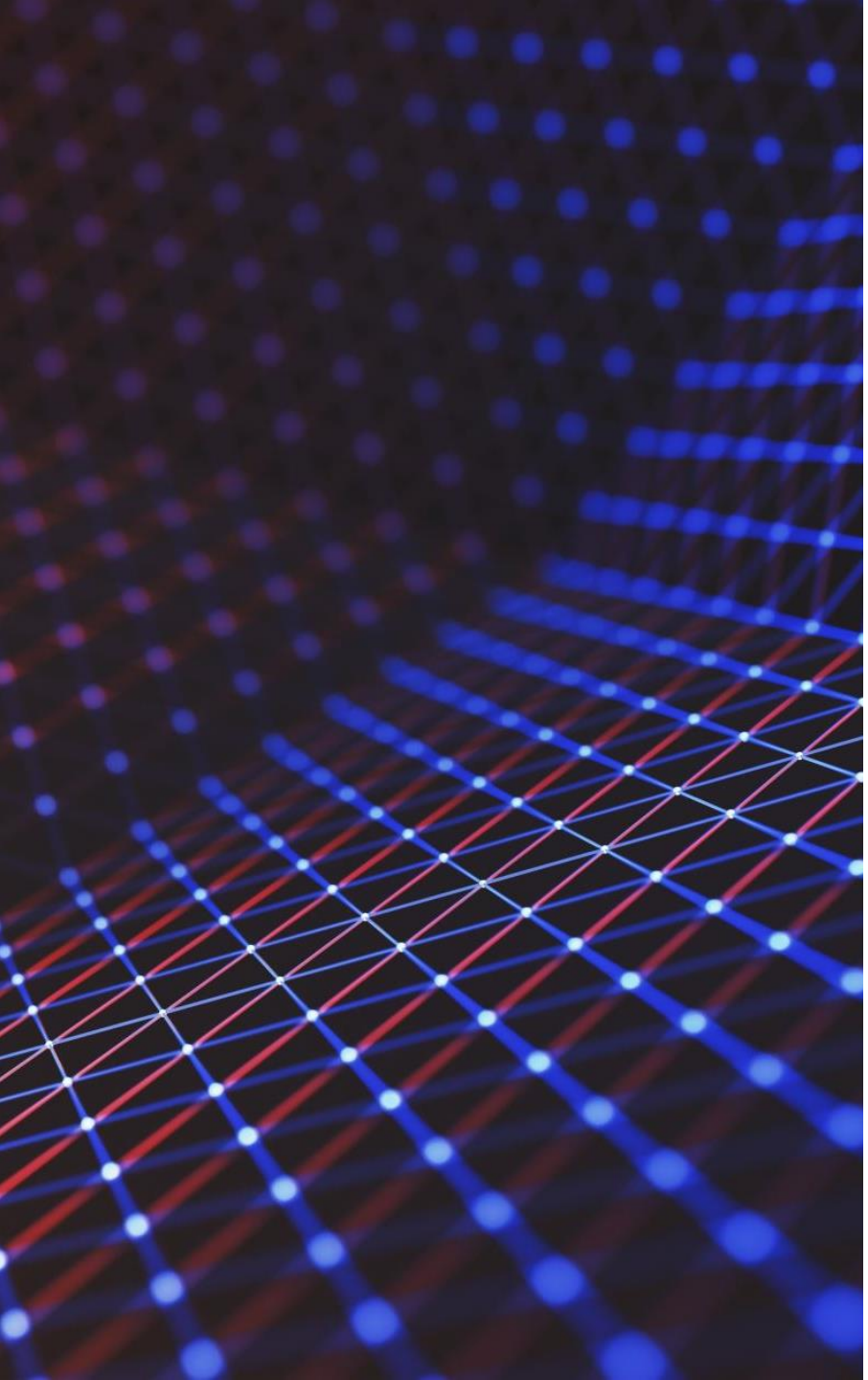
GBM (GRADIENT BOOSTING MACHINE) ALGORITHM

- *used for ML-Based Functions:*
  - Classification
  - Forecasting
  - Anomaly Detection
- = ensemble *boosting* technique combining sequentially preds of multiple weak learners (typically decision trees)
  - generalization of AdaBoosting
  - versions XGBoost and LightBoost
- *snowflake.ml.modeling* ← in Snowpark ML
  - *ensemble*.**GradientBoostingClassifier/Regressor**
  - *ensemble*.**HistGradientBoostingClassifier/Regressor**
  - *lightgbm*.**LGBMClassifier/Regressor**



SNOWFLAKE.ML.MODELING.ENSEMBLE
SNOWPARK ML NAMESPACE

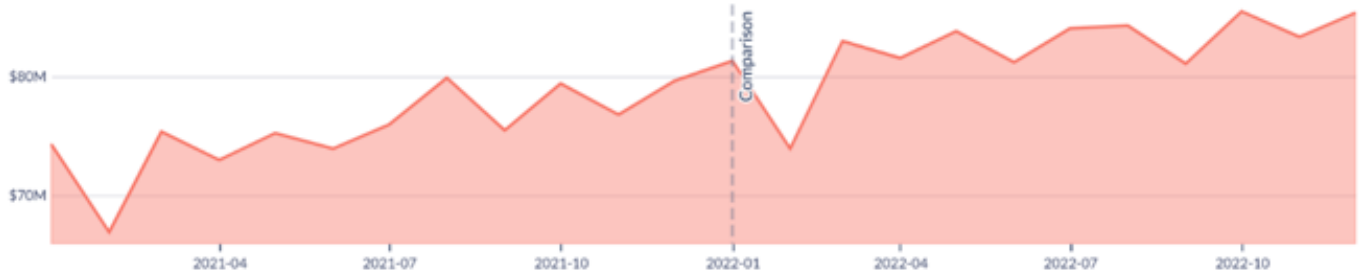
- **GradientBoostingClassifier/Regressor** = GBM
- HistGradientBoostingClassifier/Regressor = w/ histogram
- **AdaBoostClassifier/Regressor**
- **BaggingClassifier/Regressor**
- **ExtraTreesClassifier/Regressor**
- **IsolationForest**
- **RandomForestClassifier/Regressor**
- **StackingRegressor**
- **VotingClassifier/Regressor**



ENSEMBLE LEARNING

- = w/ N multiple algs (***weak learners***) → majority vote → final prediction
  - ***homogenous*** = same alg on N diff chunks of data
  - ***heterogenous*** = N diff algs for the whole data
- **(1) bagging** (Bootstrap Aggregating) = homogenous, N models+N chunks of data, *in parallel* → ***Bagging, Voting, RandomForest***
- **(2) boosting** = homogenous, N models on whole dataset, *in sequence* → ***AdaBoost, GradientBoosting, XGBoost***
- **(3) stacking** = heterogenous, N models w/ 1-2 chunks, *in parallel* → ***Stacking***

CONTRIBUTION EXPLORER

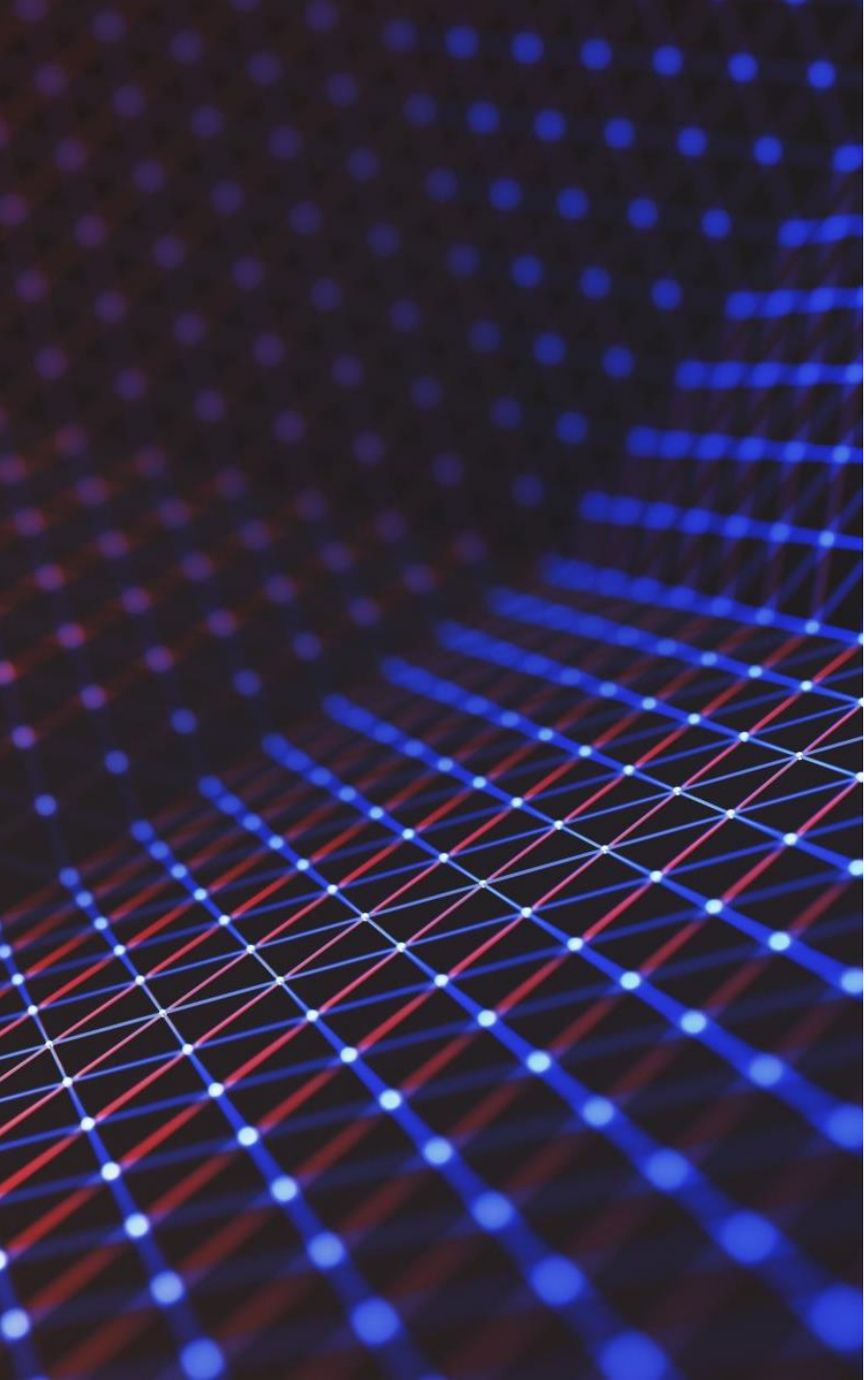


What were the key drivers that increased revenue from $906M in 2021 to $988M in 2022?
Contribution Explorer can easily analyze this for us.

Contribution Explorer Output

Insight ID	Growth Rate	Insight	Customer Count >	Customer Count <	Region =	Region !=	Prc
1	74.56	"Customer Count <= 951.5","Customer Count > 583.0","not Product Type = <oth...	583	951	Southwest	null	null
2	60.67	"Customer Count <= 719.0","not Product Type = Mobiles","Store Region = Midw...	null	719	Midwest	null	null
3	53.19	"Customer Count > 956.0","not Product Type = Photography","not Store Region ...	956	null	null	Midwest	null
4	51.33	"Customer Count > 956.0","not Product Type = Entertainment","not Store Regio...	956	null	null	Midwest	null
5	51.26	"Customer Count > 956.0","not Product Type = Mobiles","not Store Region = Mi...	956	null	null	Midwest	null
6	33.33	"Customer Count > 573.0","not Product Type = Arts & Entertainment","Store Re...	573	null	Southwest	null	null
7	32.39	"Customer Count > 956.0","not Store Region = Midwest"	956	null	null	Midwest	null
8	28.29	"Customer Count > 956.0","not Product Type = <other>","not Store Region = Mi...	956	null	null	Midwest	null
9	26.50	"Customer Count > 554.5","not Product Type = Music","Store Region = Southwe...	554	null	Southwest	null	null
10	25.53	"Customer Count > 951.5","not Product Type = Arts & Entertainment","not Store...	951	null	null	Midwest	null
11	25.53	"Customer Count > 952.0","not Product Type = Arts & Entertainment"	952	null	null	null	null
12	24.15	"Customer Count > 952.0","not Product Type = Mobiles","not Store Region = Mi...	952	null	null	Midwest	null
13	23.13	"Customer Count <= 719.0","Store Region = Midwest"	null	719	Midwest	null	null
15	21.99	"Customer Count > 951.5","not Product Type = Entertainment"	951	null	null	null	null

1,934 Rows – 19 Columns



CONTRIBUTION EXPLORER

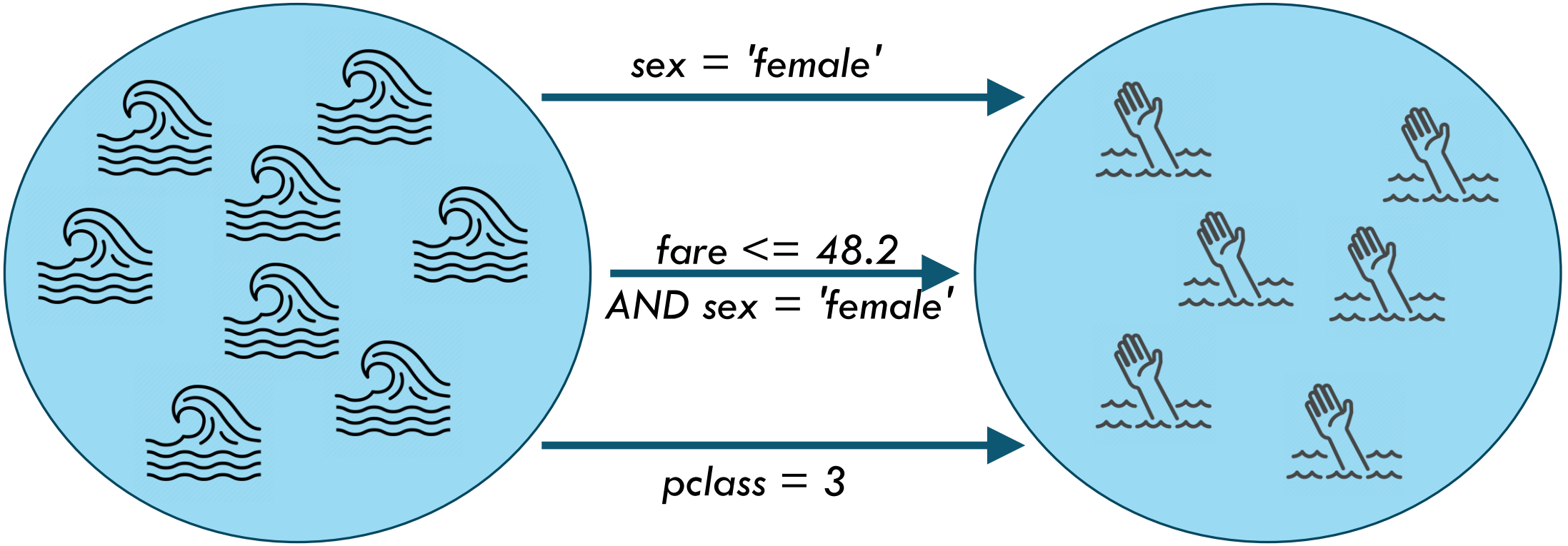
- = determines the factors that influence a value of interest
- **RCA (Root Cause Analysis)** = extract root causes from datasets w/ large number of dims
- finds most important dims in a dataset + builds segments from those dims + detects which of those segments influenced the metric
- runtime scales with the number of dims and the cardinality of those dims
- cardinality of categorical dims is automatically reduced when their cardinality exceeds *5*
- ...NOT A TIME-SERIES FUNCTION!

CONTRIBUTION EXPLORER FUNCTION CALL

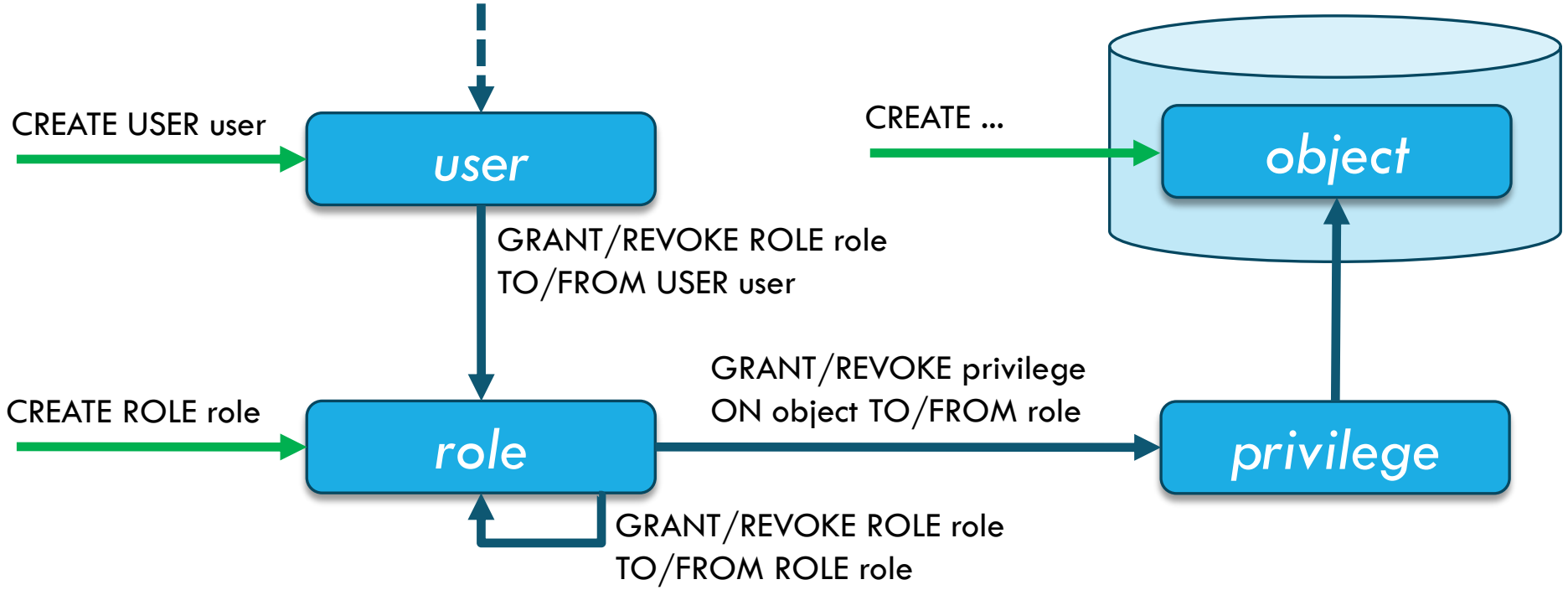
```
SELECT * FROM input,
  TABLE(TOP_INSIGHTS(
    input.categorical_dimensions,
    input.continuous_dimensions,
    input.metric,
    input.label)
```

(TABLE)	
contributor	ARRAY
metric_control	FLOAT
metric_test	FLOAT
surprise	FLOAT
relative_change	FLOAT
growth_rate	FLOAT
expected_metric_test	FLOAT
overall_metric_control	FLOAT
overall_metric_test	FLOAT
overall_growth_rate	FLOAT
new_in_test	BOOLEAN
missing_in_test	BOOLEAN

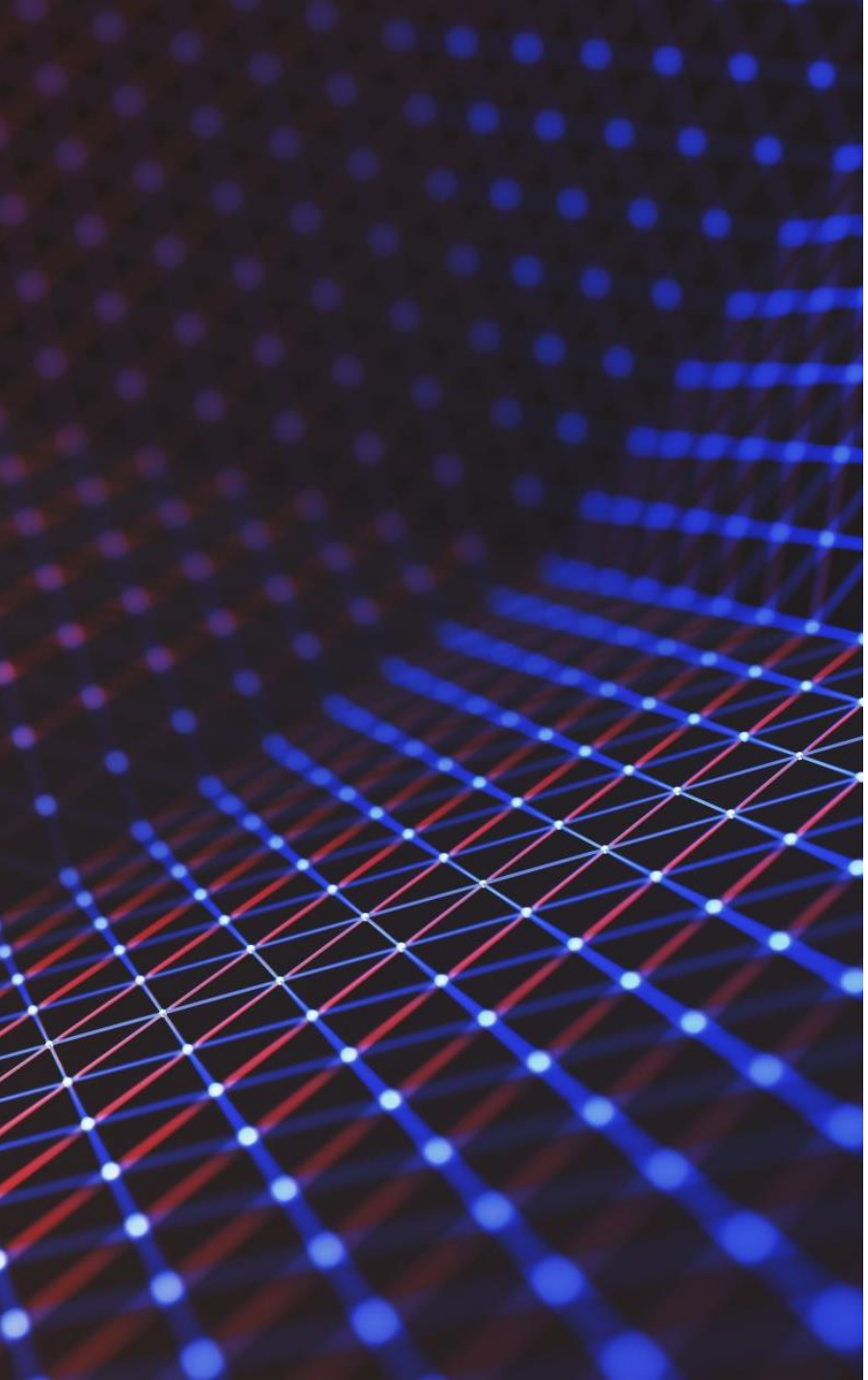
CONTRIBUTION FACTORS TO SURVIVAL ON TITANIC



SNOWFLAKE ACCESS CONTROL FRAMEWORK

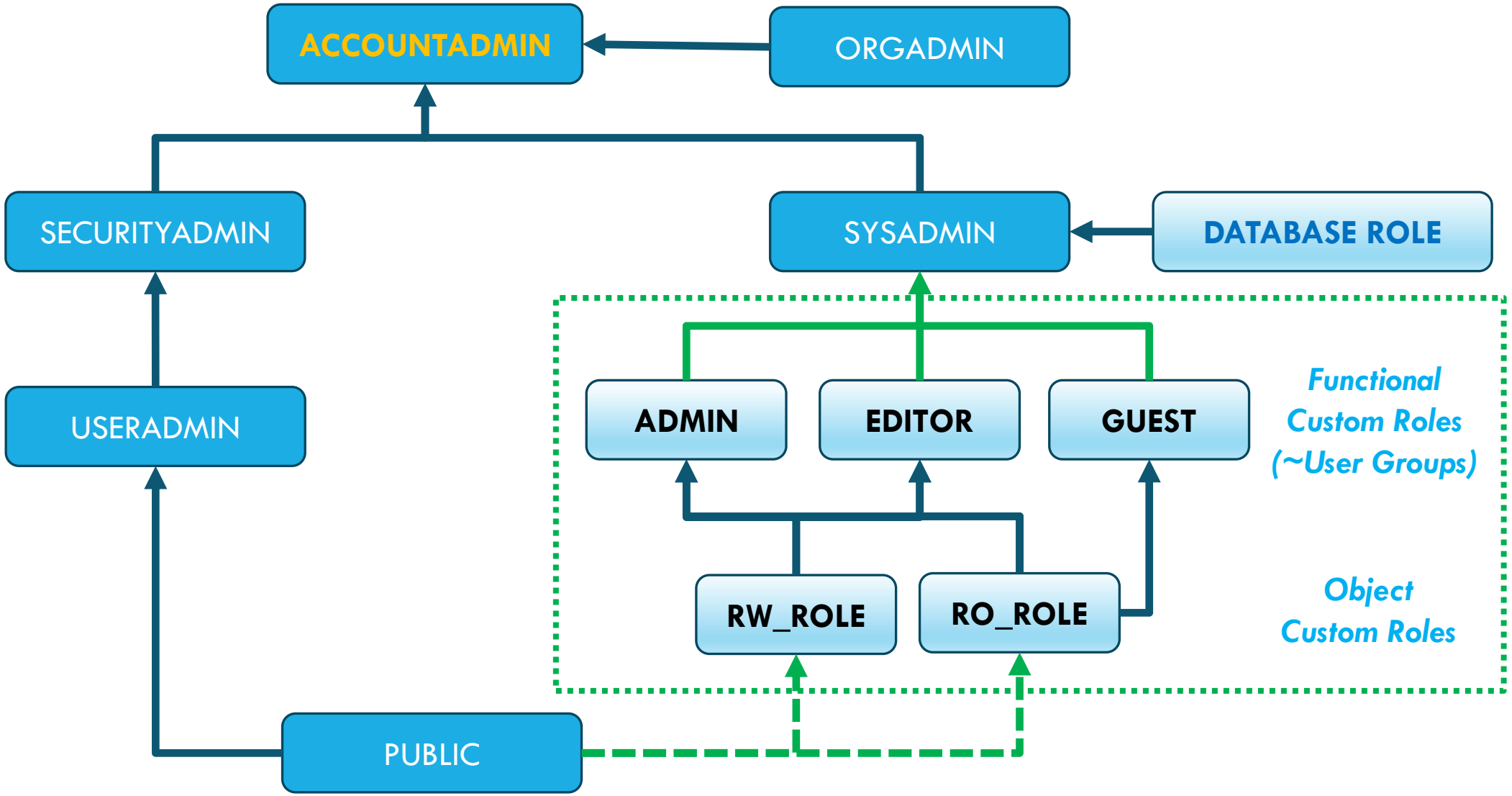


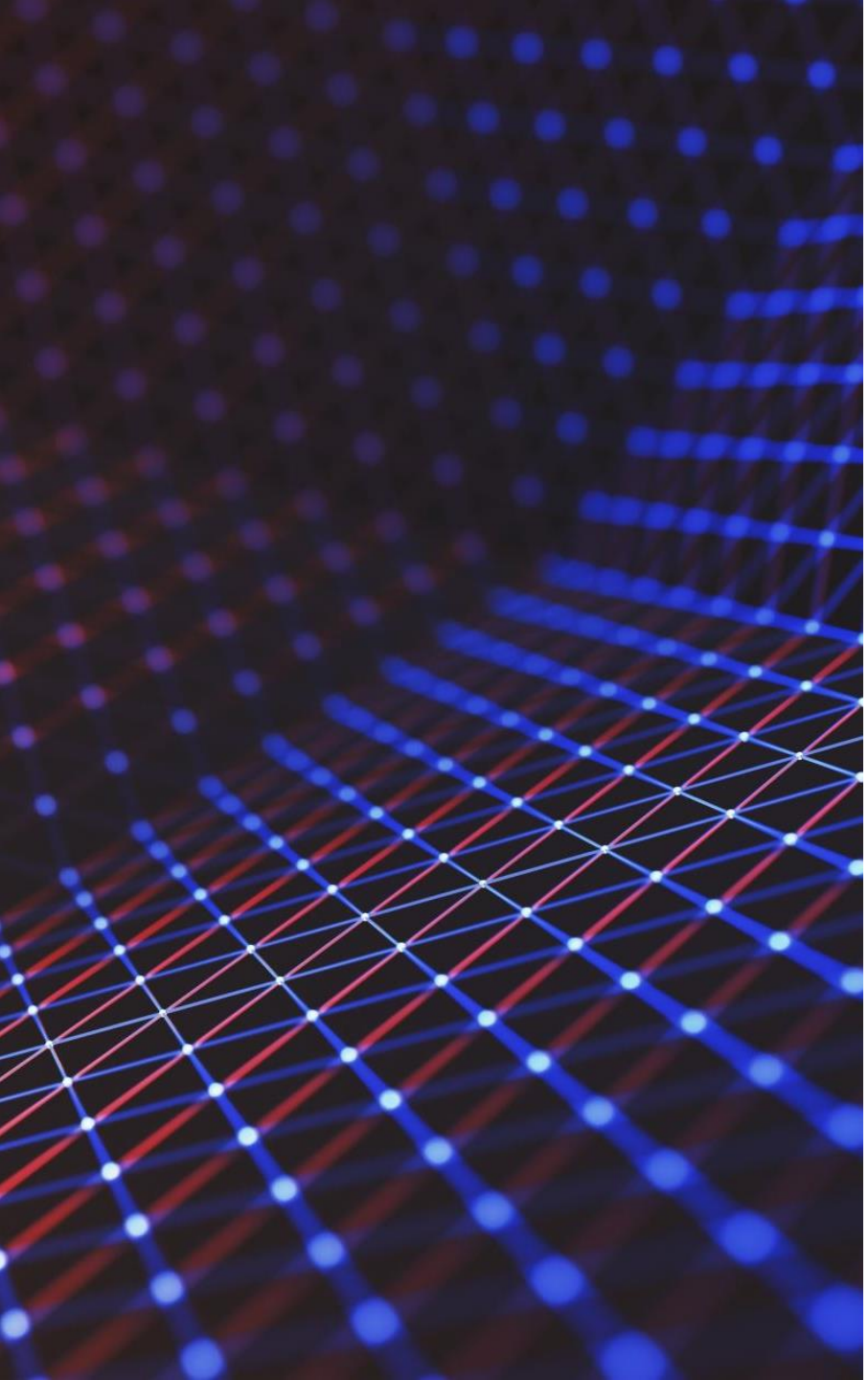
- **DAC (Discretionary Access Control)** - objects ← grant access ← owners
- **RBAC (Role-Based Access Control)** - objects ← privileges ← roles ← users



ROLE TYPES

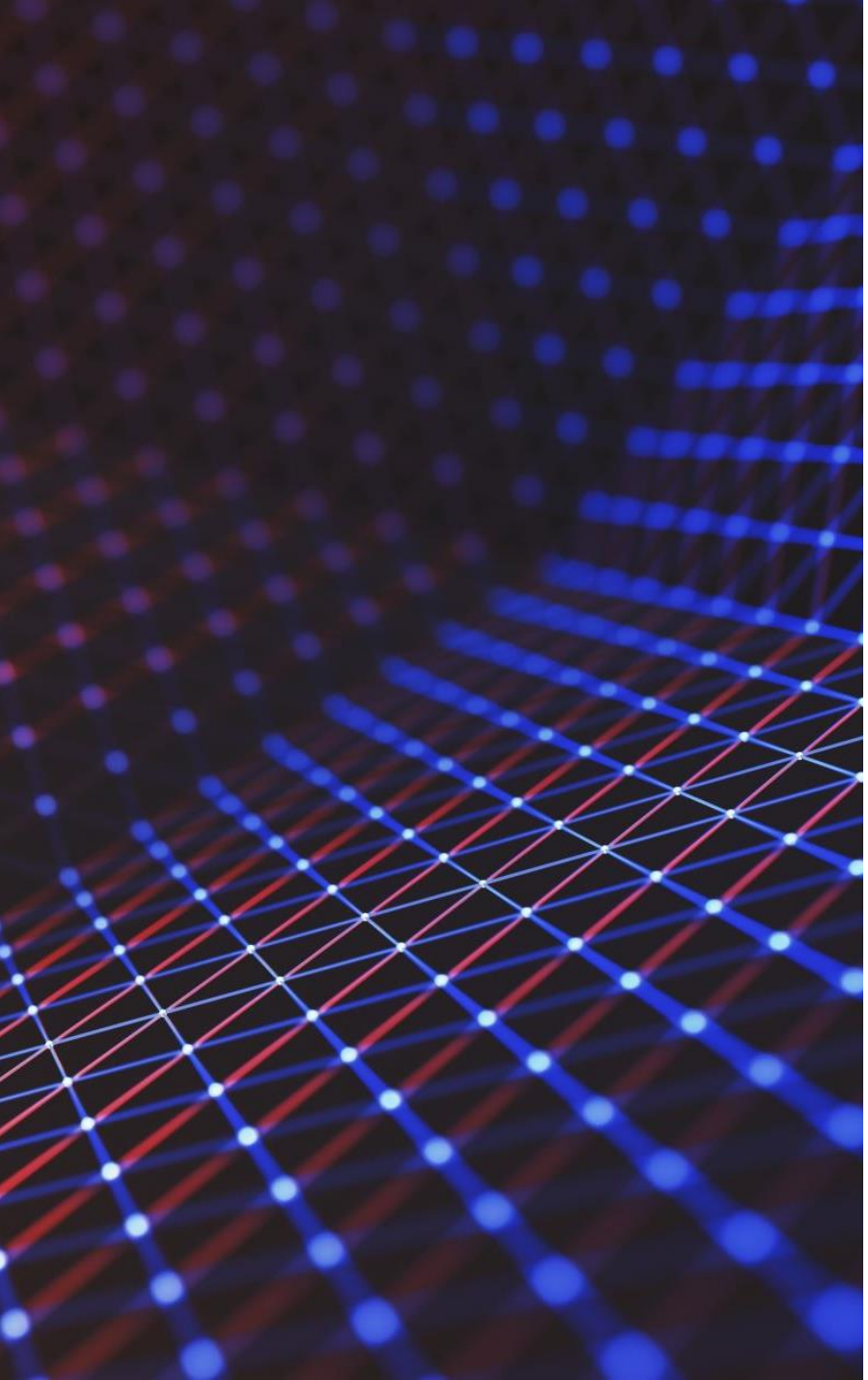
- global → ***account roles*** = for privs to any object
  - system-defined roles = built-in, cannot change
  - custom roles = user-def, PUBLIC, use SYSADMIN
  - inherited roles = in the role hierarchy
- session → ***active roles*** = current context session role
  - primary role = current role in active user session
  - secondary roles = can also be activated in session
- database → ***database roles*** = to objects in a single db
- class → ***instance roles*** = for access to instance methods
  - class roles = for privs on the class methods
- app → ***application roles*** = for Snowflake Native apps


ACCOUNTADMIN
ORGADMIN
SECURITYADMIN
SYSADMIN
DATABASE ROLE
USERADMIN
ADMIN
EDITOR
GUEST
Functional
Custom Roles
(~User Groups)
RW_ROLE
RO_ROLE
Object
Custom Roles
PUBLIC



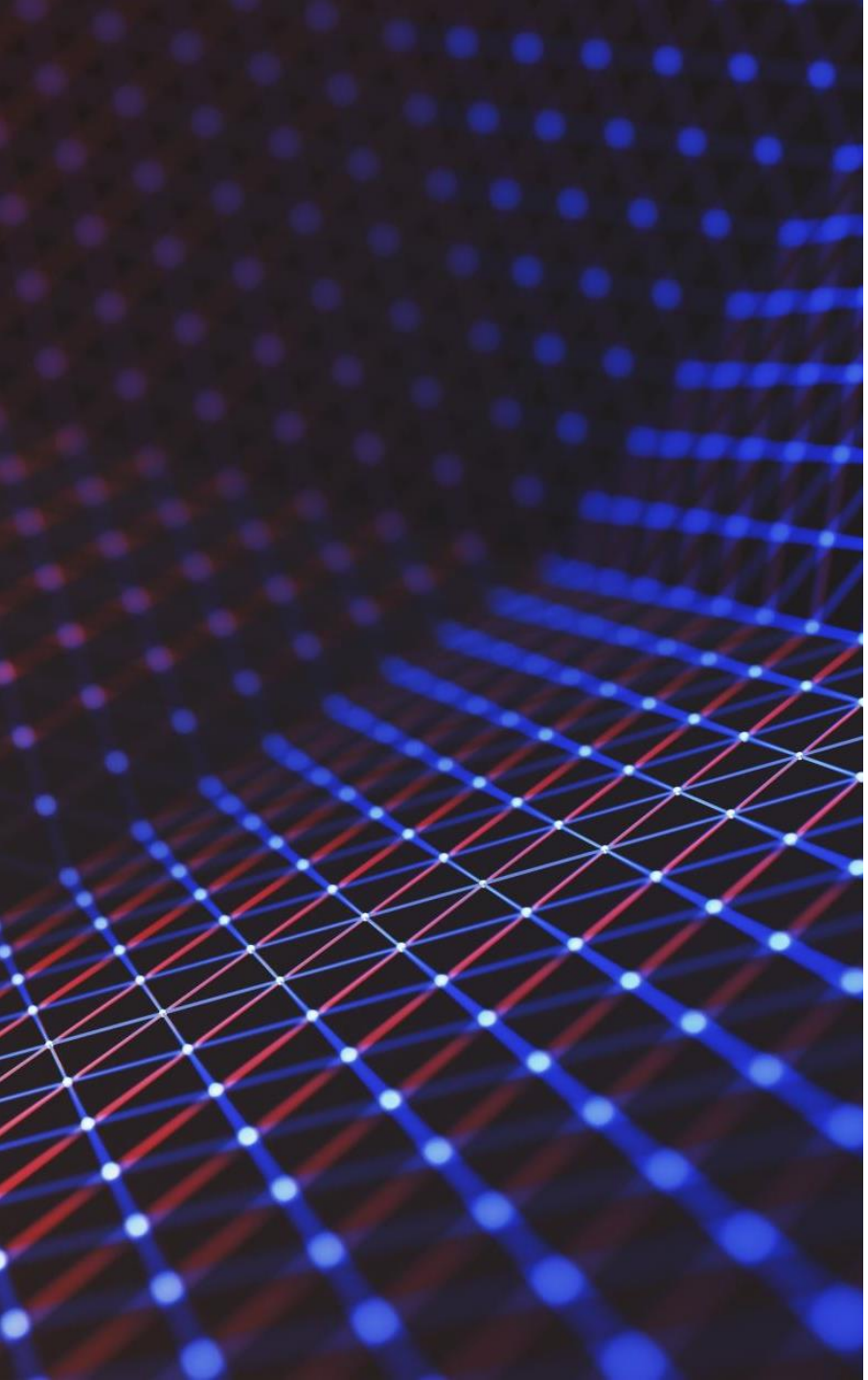
SYSTEM-DEFINED ROLES

- **ORGADMIN** = separate, to manage orgs
- **ACCOUNTADMIN** = top-level
- **SECURITYADMIN** = to manage roles/privileges
- **SYSADMIN** = to create objects
- **USERADMIN** = to create users
- **PUBLIC** = assigned to any other role



BUILTIN INSTANCE ROLES FOR CLASSIFICATION

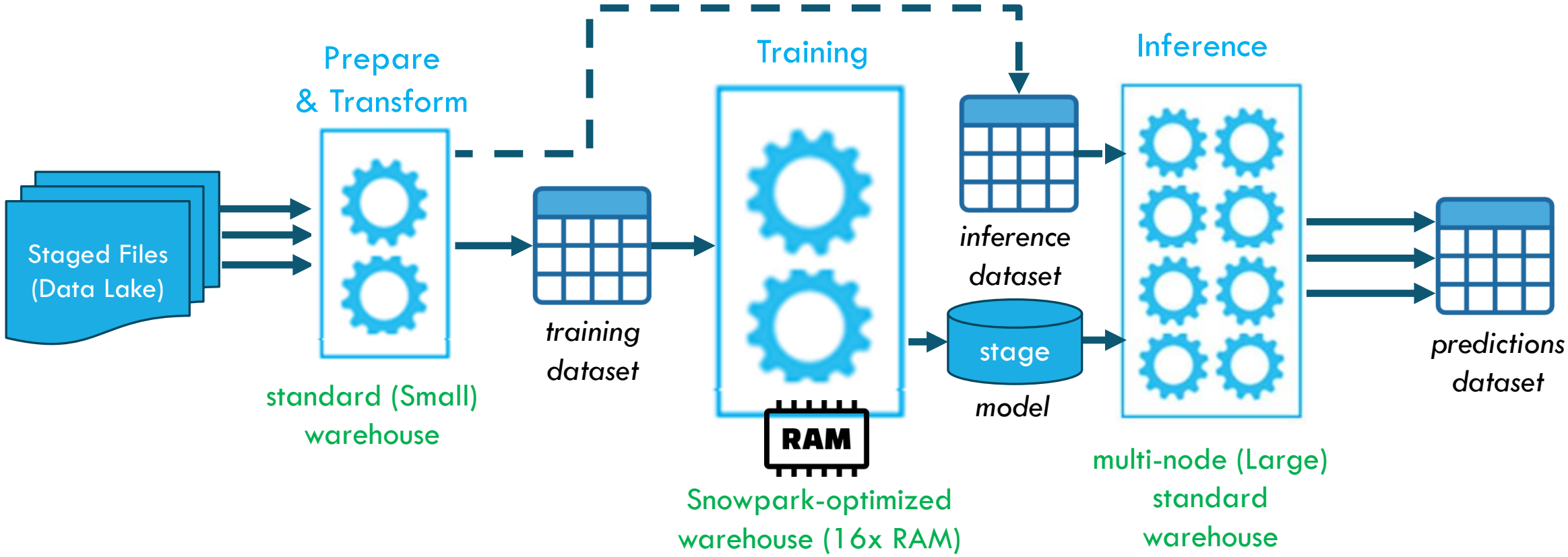
- model!**MLADMIN**
  - can create classification models
  - can grant access privileges to own model instances
  - should assign to custom ***analyst*** role
  - grant CREATE SNOWFLAKE.ML.CLASSIFICATION priv
  - should have access to underlying table/query/WH
- model!**MLCONSUMER**
  - cannot create/train classification models
  - cannot access metrics and other internal data
  - can only call PREDICT
  - should assign to custom ***consumer*** role (by analyst)
  - grant model!MLCONSUMER role (after model create)
  - should have access to underlying table/query/WH



BUILTIN INSTANCE ROLE FOR FORECASTING/ANOMALY DETECTION

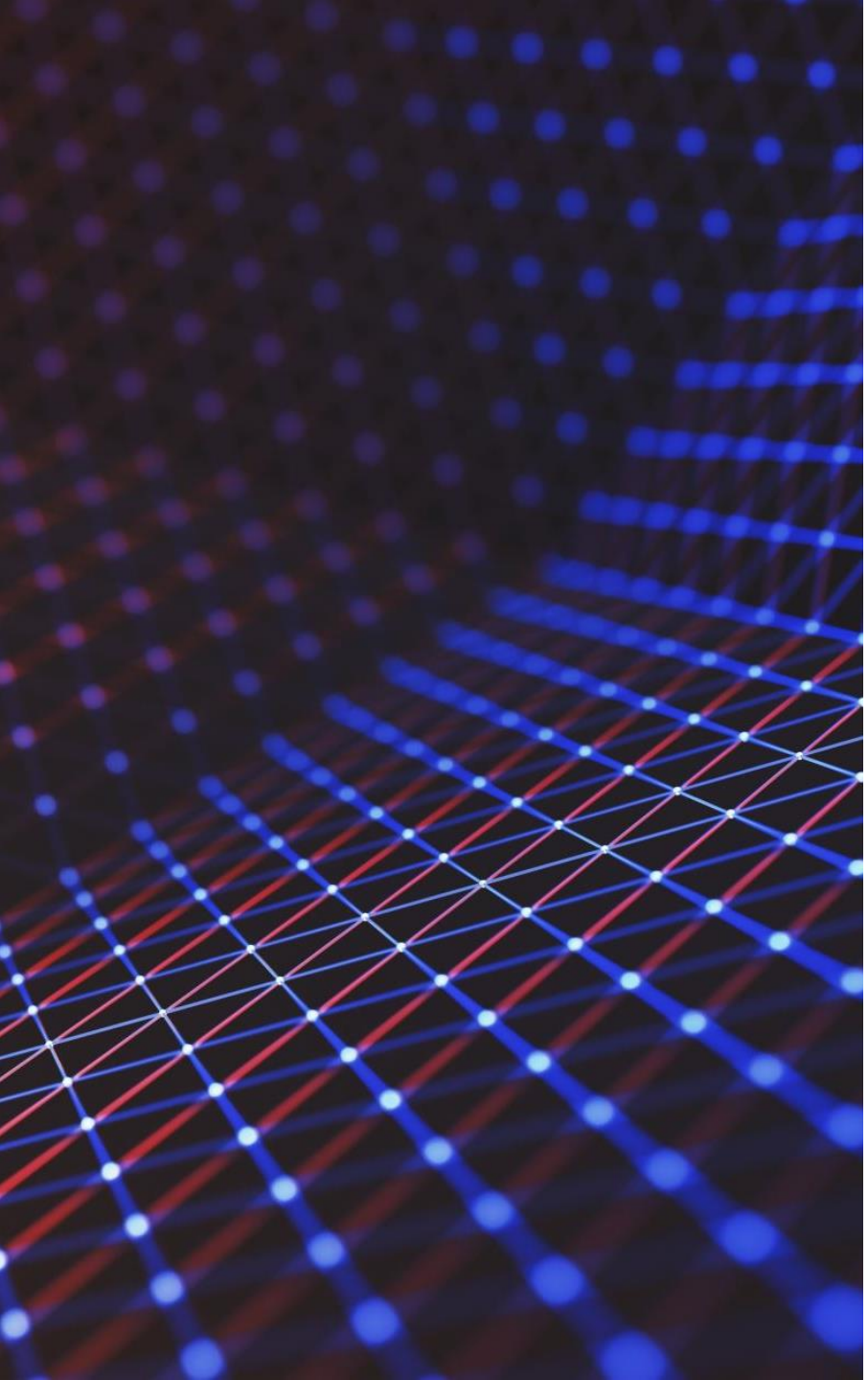
- owner
  - create/own/train/drop models + call any other method
  - should assign to a custom ***analyst*** role
  - grant CREATE SNOWFLAKE.ML.<class> priv
  - should have access to underlying table/query/WH
- model!**USER**
  - can create/own/train/drop models
  - can call any method on existing instance
  - should assign to a custom ***consumer*** role
  - grant model!USER role (after model create)
  - should have access to underlying table/query/WH

SELECTION OF WAREHOUSES



VIRTUAL WAREHOUSE (COMPUTE) CREDITS

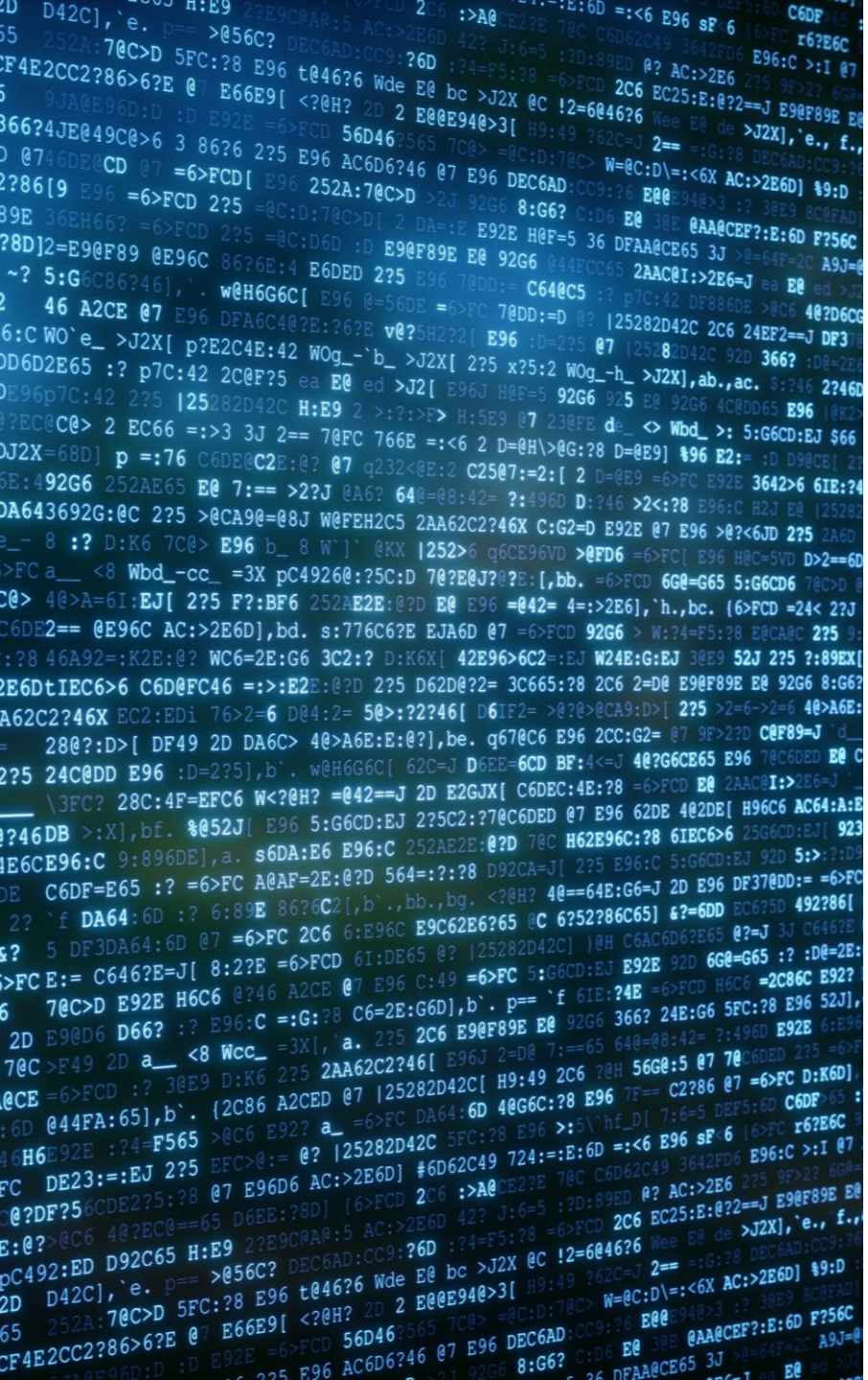
Type of Virtual Warehouse	X-Small	Small	Medium	Large	X-Large	2X-Large	3X-Large	4X-Large	5X-Large	6X-Large
Standard	1	2	4	8	16	32	64	128	256	512
Snowpark-optimized	n/a	n/a	6	12	24	48	96	192	384	768



COMPUTE COSTS (WAREHOUSES)

- *training* → consumes more compute than prediction
- *prediction* → cost can accumulate, with repeated model use

- **Standard Virtual Warehouse** (dedicated)
  - small, for single TS w/ few rows + exogenous vars
  - small, if no vars + <5M rows
  - larger, for multiple TS or model inference/predictions
  - small or dep on dataset size, for Classif/Contrib
  - small, for Prepare & Transform (w/ concurrency!)
- **Snowpark-optimized Virtual Warehouse**
  - for single TS w/ 5M+ rows or many vars
  - for multiple TS (10/100/1000)
  - for large dataset in Classif



STORAGE COSTS (MODELS)

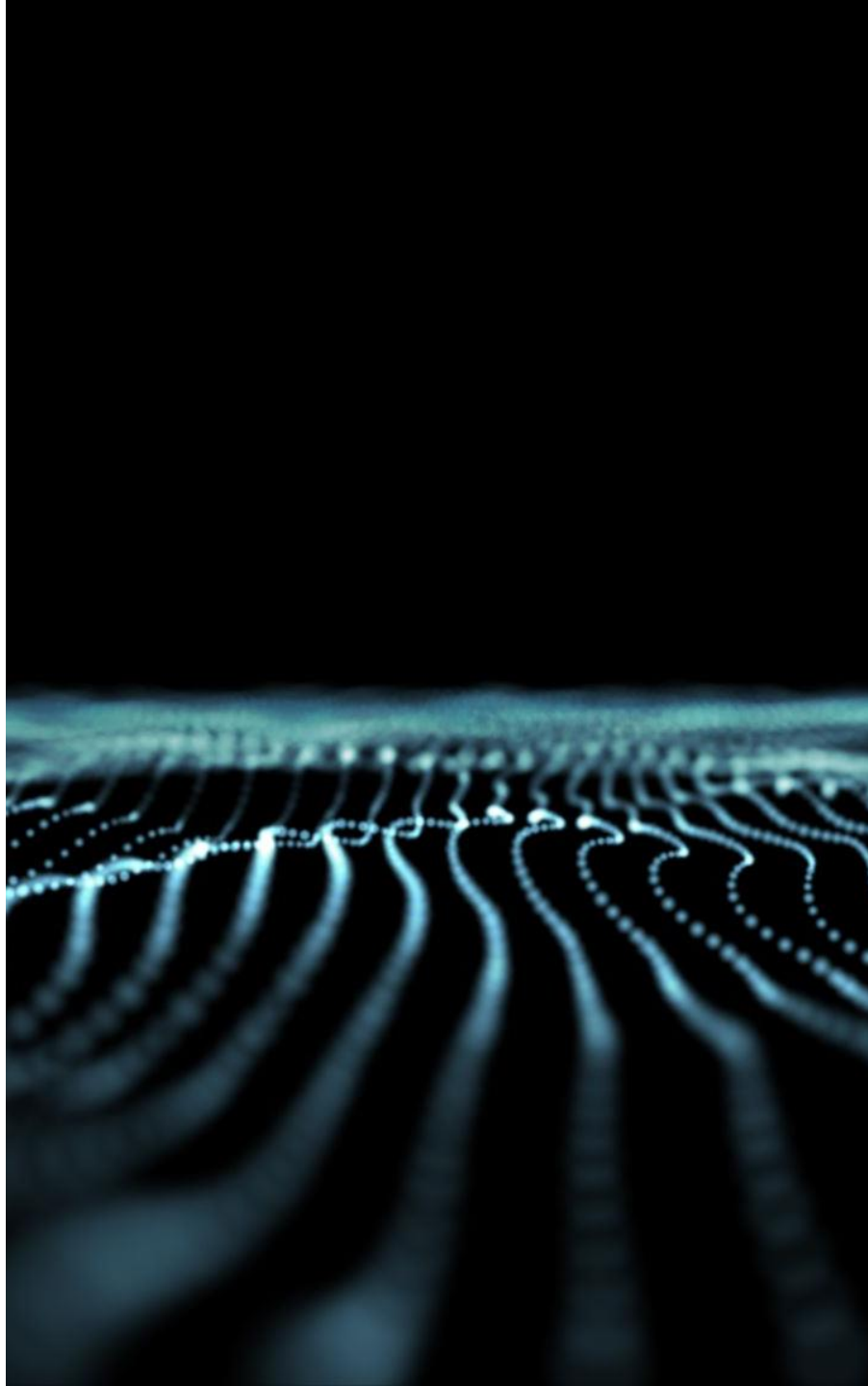
- for ML model instances created during the training step
- ACCOUNT_USAGE.**TABLES/STAGES** views → by instance_id (NULL db/schema)
- delete unused/obsolete models, to reduce storage costs

LLM FUNCTIONS

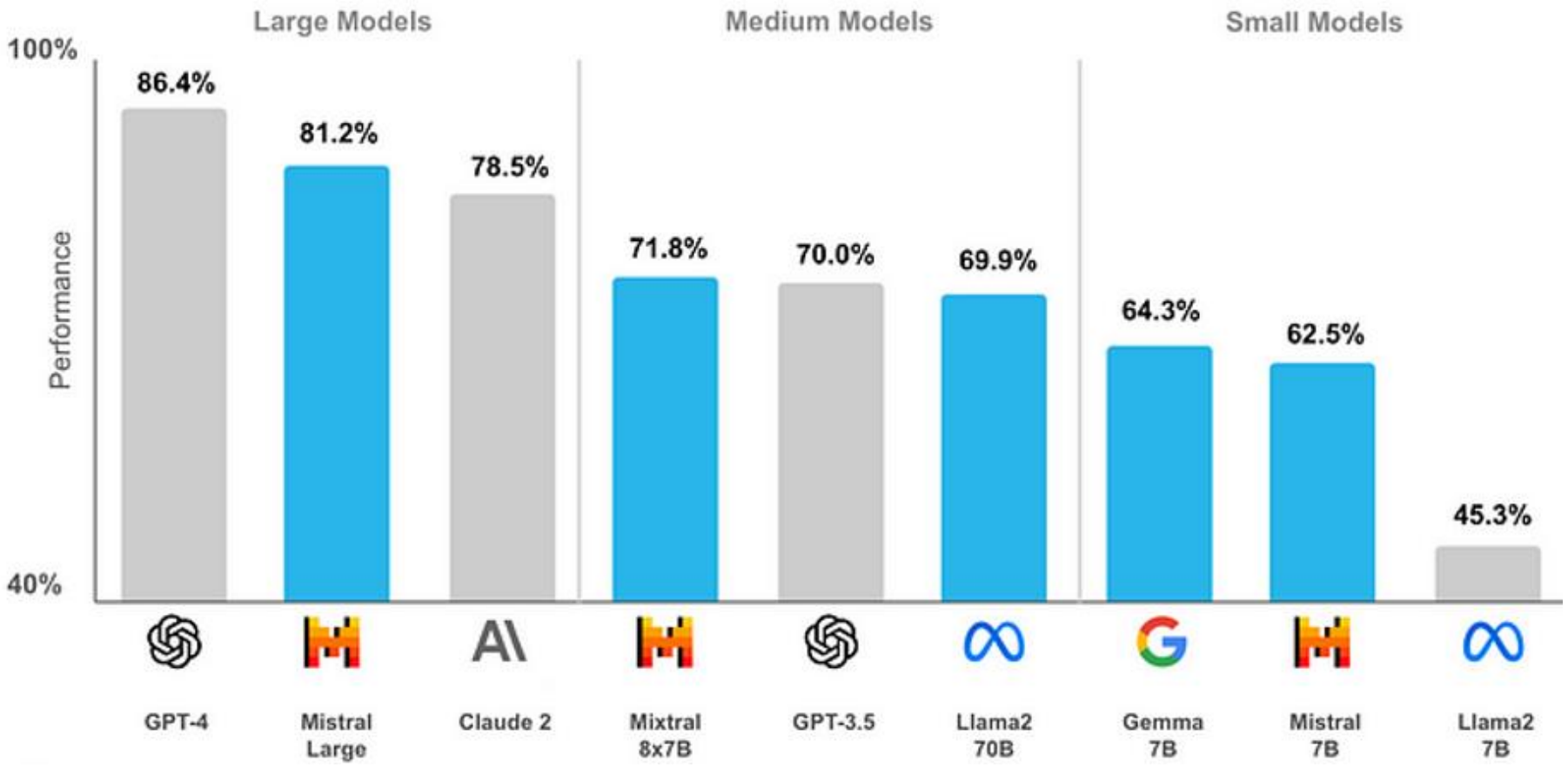
- COMPLETE
- EXTRACT_ANSWER
- SENTIMENT
- SUMMARIZE
- TRANSLATE

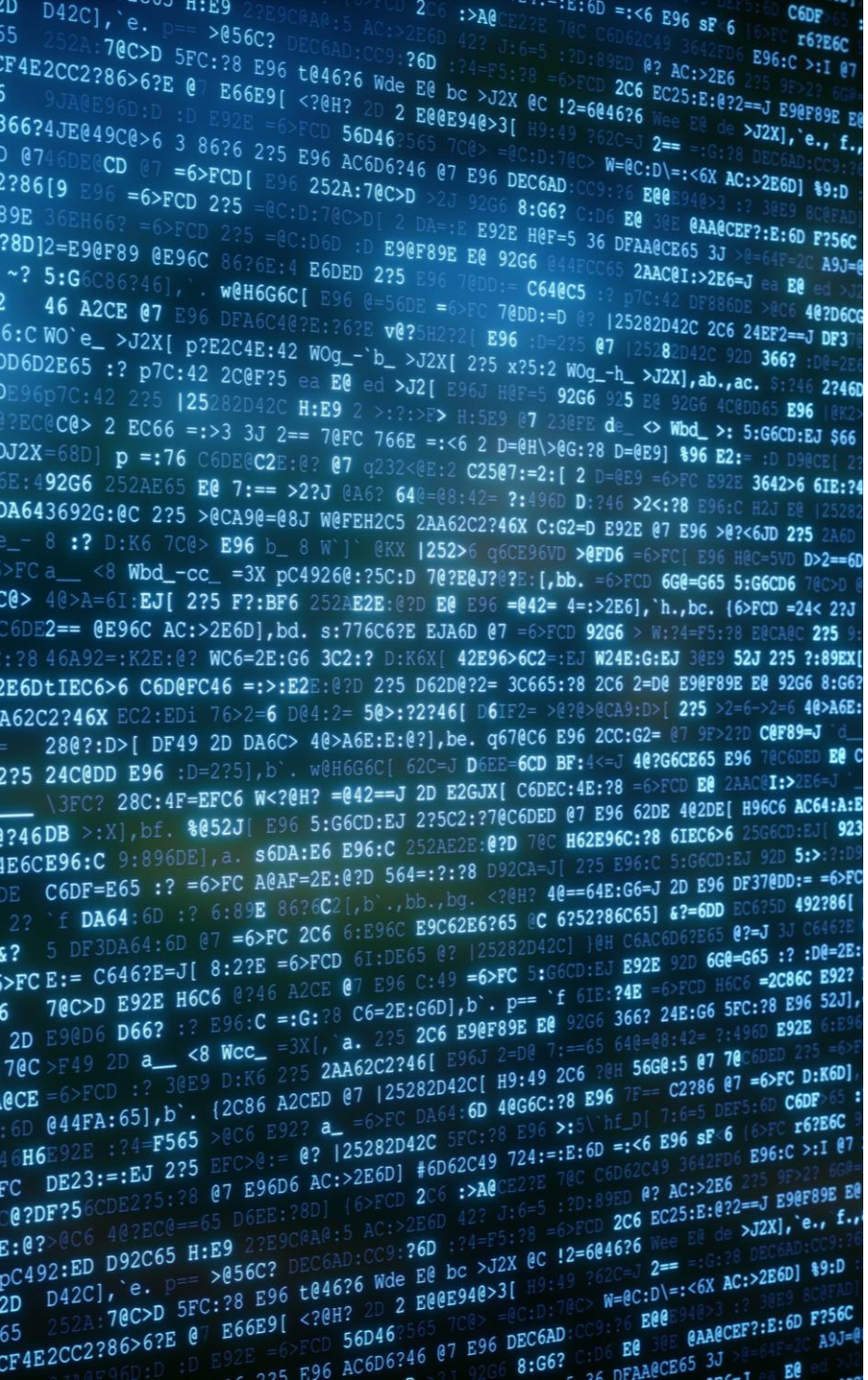
ALL LLM FUNCTIONS

- = as secure serverless functions (UDFs) in the ***SNOWFLAKE.CORTEX*** schema
- **COMPLETE** = ~generic OpenAI/ChatGPT Completion API + can select model
- **EXTRACT_ANSWER** = extract info from unstructured or semi-structured data
- **SENTIMENT** = detect mood/tone of text
- **SUMMARIZE** = automatically get a synopsis of text
- **TRANSLATE** = translate docs to other languages



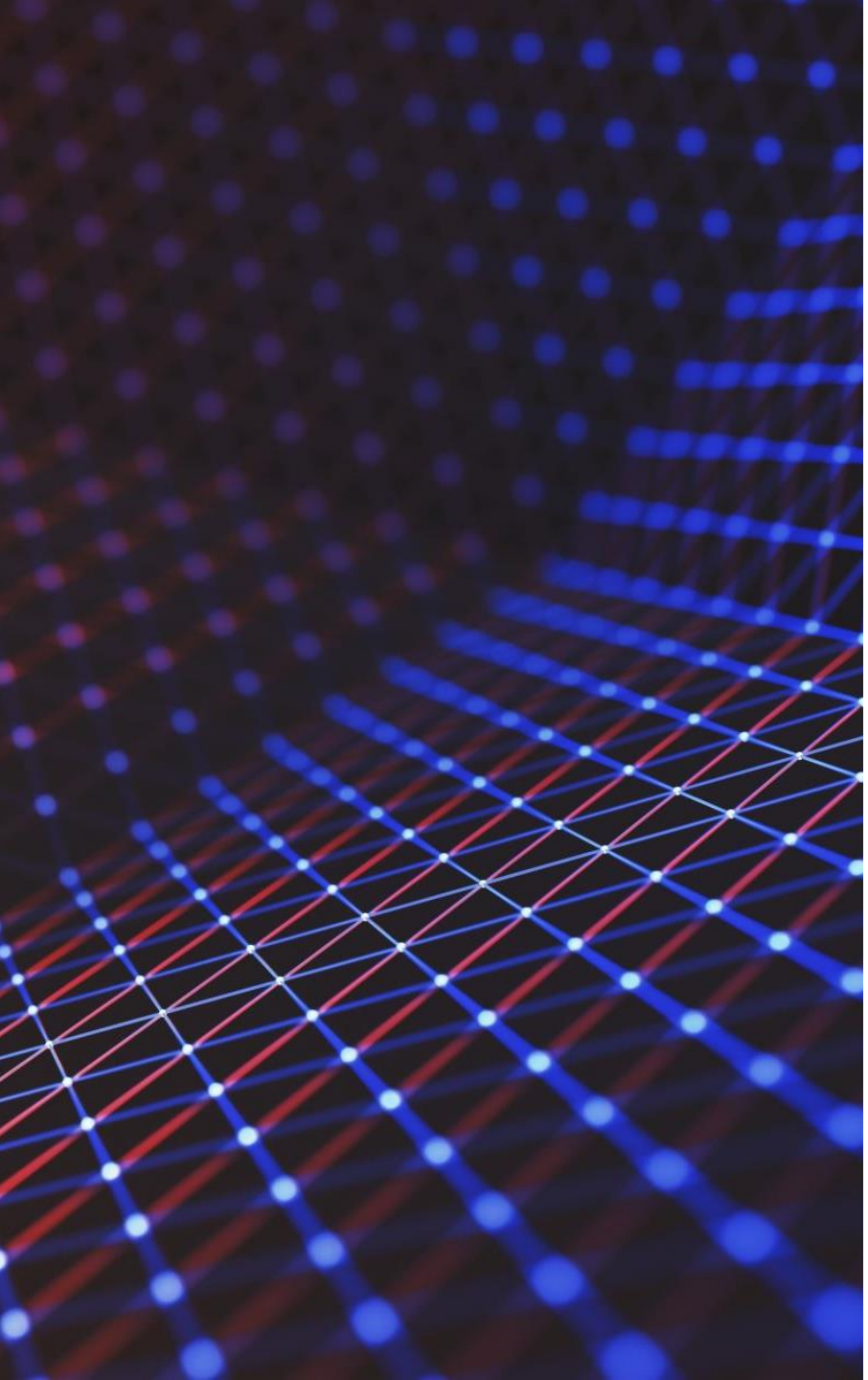
LLMS
(LARGE LANGUAGE MODELS)





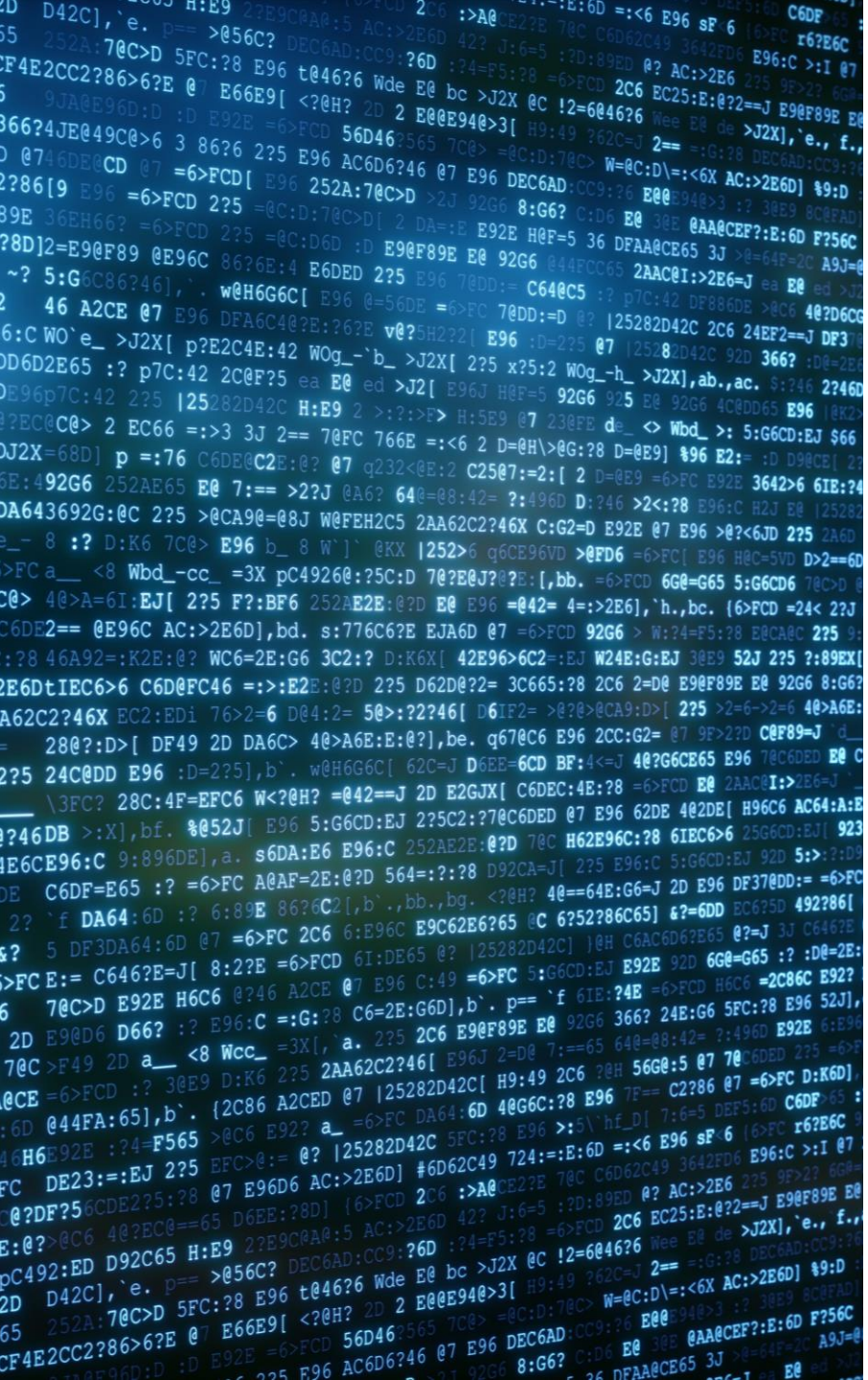
SNOWFLAKE-HOSTED LLMS

- 80%+ perf models: by Mistral (Large, 8x7B, 7B), Meta (Llama2 70B), Google (Gemma 7B)
- LLMs hosted in Snowflake → more security protection & data governance
- specialized functions → no prompt engineering req
- no separate APIs or conns → just SQL function calls
- unlike w/ ChatGPT, customer data is never used to train or fine-tune the hosted models
- serverless → hosted models are never downloaded on the client side
- can use to build own RAG, w/ new VECTOR data type



DATA SCIENCE
A FEW HISTORIC MILESTONES

- statistics
- English Query
- Data Mining
- Machine Learning
- Deep Learning
- Generative AI



DEEP LEARNING NEURAL NETWORKS (NN)

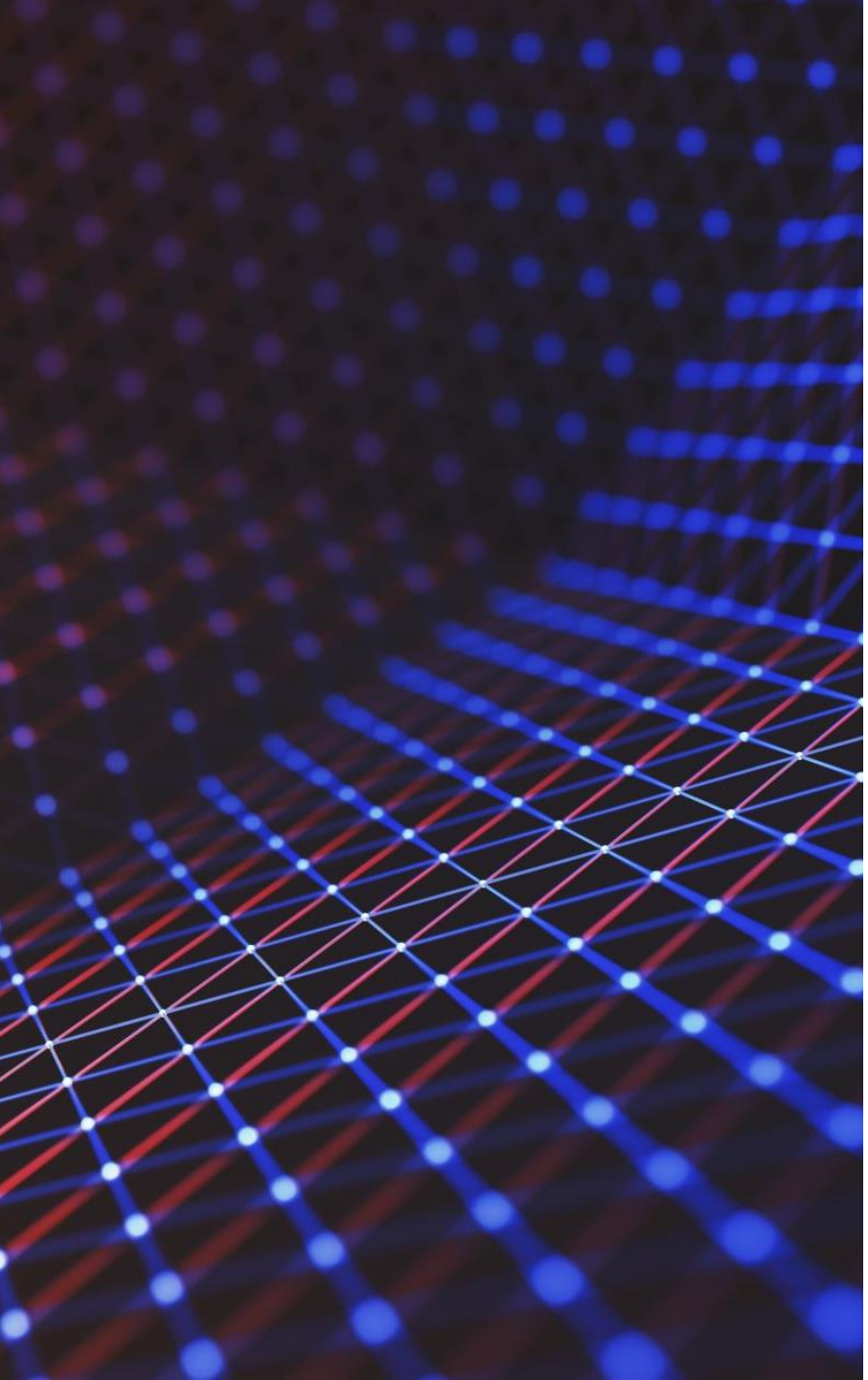
- **ANN (Artificial NN)**
- **DNN (Deep NN)** = ANN w/ multiple hidden layers
- **FNN (Feed-forward NN)** = w/ no memory of any previous layer
- **MLP (Multi-Layer Perceptron)** = FNN w/ input+hidden+output layers
- **CNN (Convolutional NN)** = FNN w/ learnable weights/biases, for images/videos
- **RNN (Recurrent NN)** = w/ feedback output → input again
- **LSTM (Long Short-Term Memory)** = probabilistic RNN, w/ memory cell

CNN
(Convolutional NN)

- for spatial data (images) → ideal for images/video processing
- more powerful than RNN
- fixed size inputs → fixed size outputs
- FFN w/ variations of MLPs using minimal amounts of preprocessing
- use connectivity pattern between neurons → inspired by the org of animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field

RNN
(Recurrent NN)

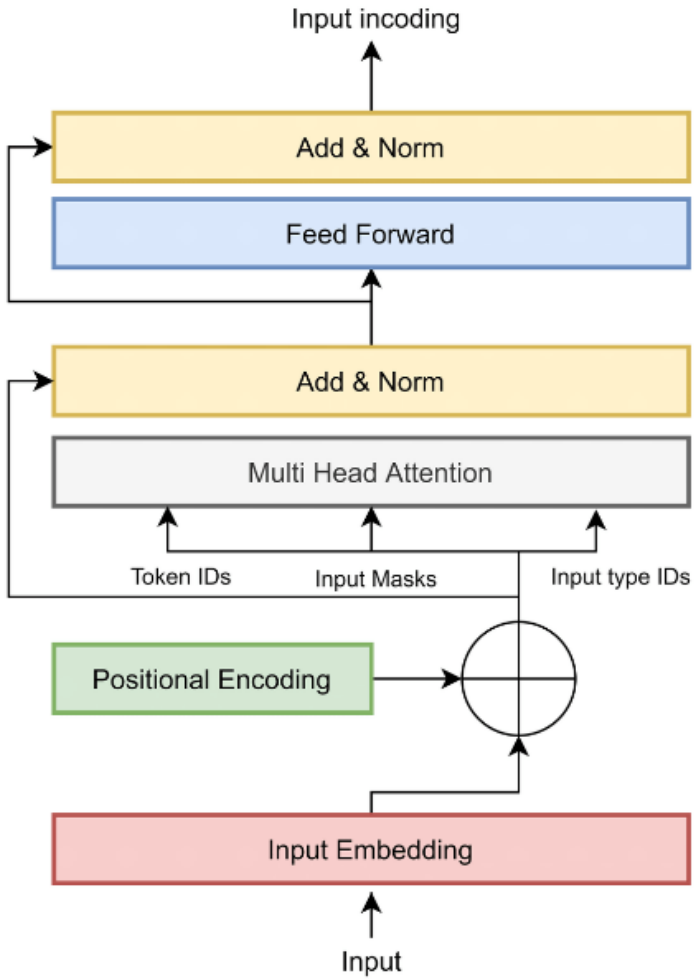
- for temporal (sequential) data → ideal for text/speech analysis
- less feature compatibility
- arbitrary input/output lengths
- can use internal memory to process arbitrary sequences of inputs
- use time-series info → what a user spoke last will impact what he/she will speak next



GENERATIVE AI

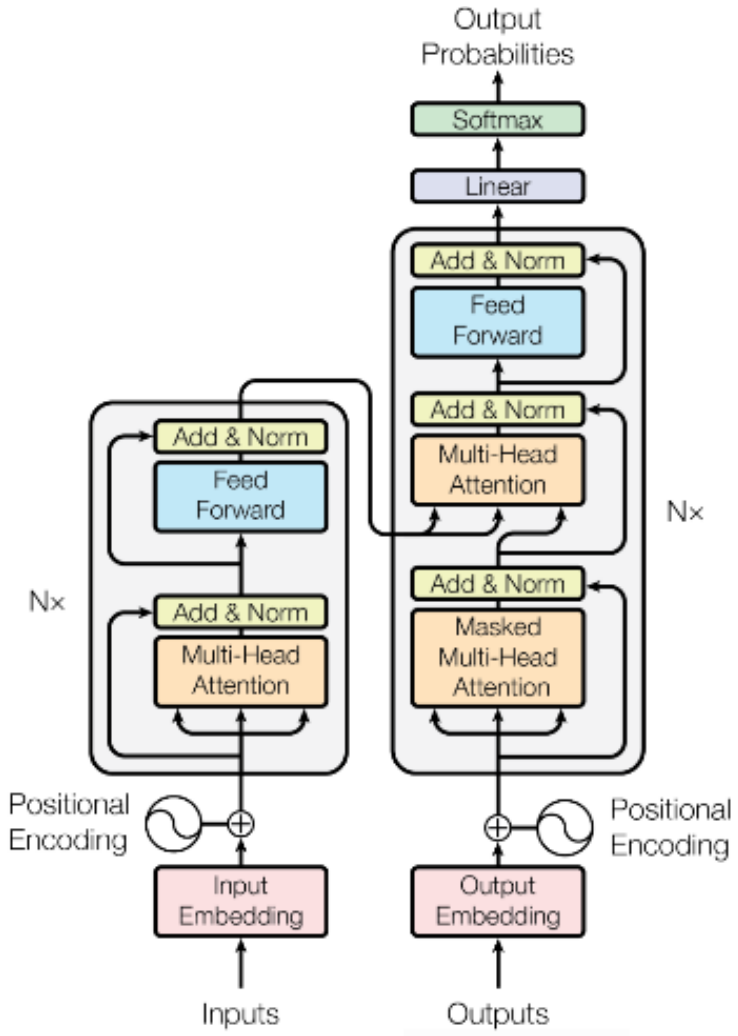
- = input text (prompts) → generated text/images/media (art/code/fashion)
- **NLP (Natural Language Processing)**
  - ***tokenisation*** = encode text (as tokens) → into numbers
  - ***embeddings*** = ~tokens, to capture semantics as vectors
  - ***vector store*** = data store for embeddings
- **Transformers** = DL model for NLP/CV, 2017+ at Google Brain, w/ "self-attention"
- **LLMs (Large Language Models)** = input text (prompts) → generated output text (can retain conversation state → allow for follow-up questions)
- **GANs (Generative Adversarial Networks)** = 2014+, w/ Generator (creator) + Discriminator (critic) → images

BERT ENCODER



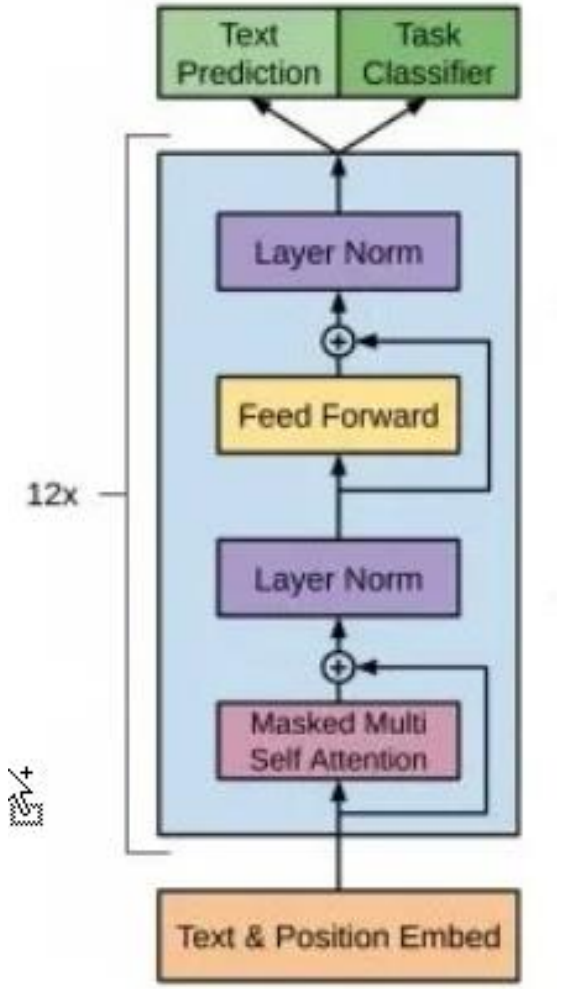
- sentence classification
- named entity recogn.
- extractive Q&A

T5 TRANSFORMER (ENCODER-DECODER)



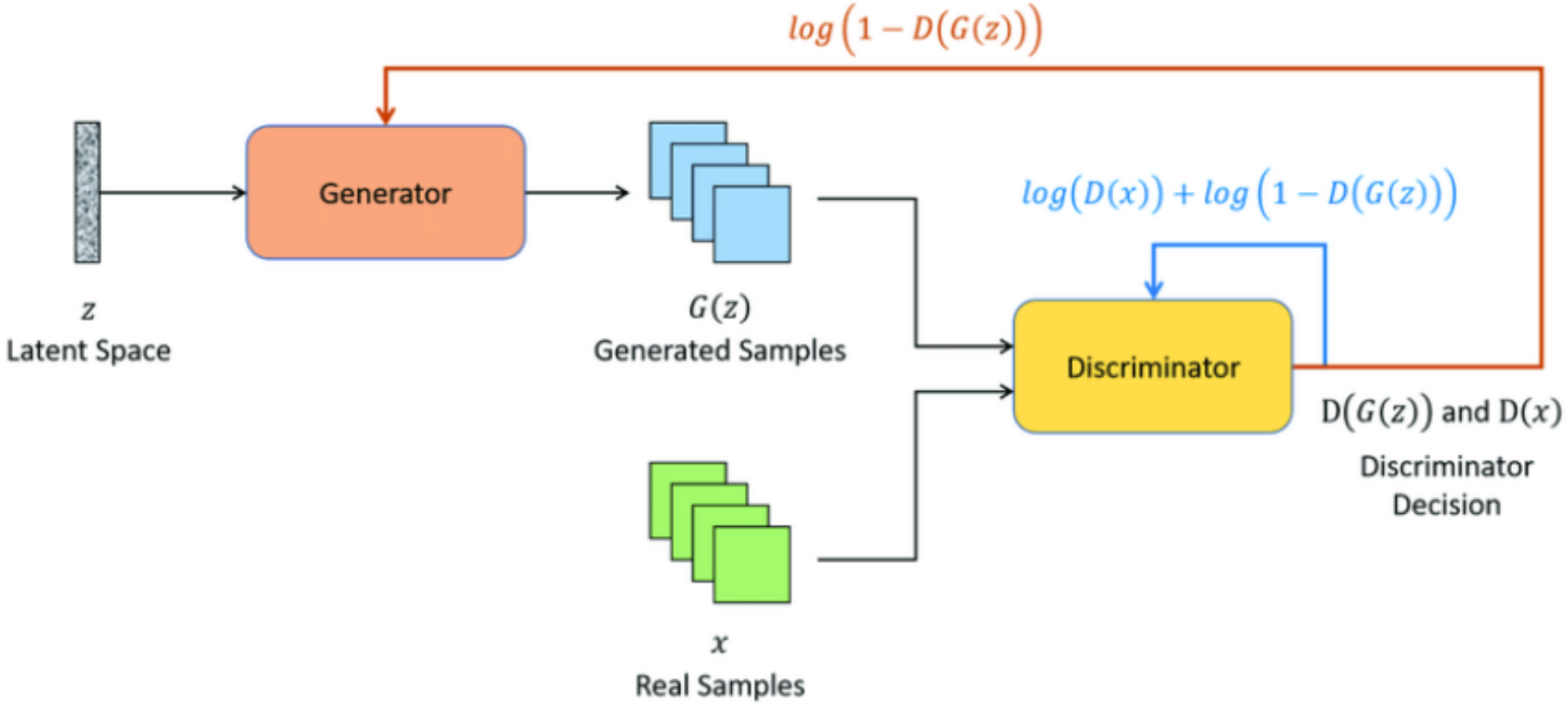
- summarization
- translation
- Q&A

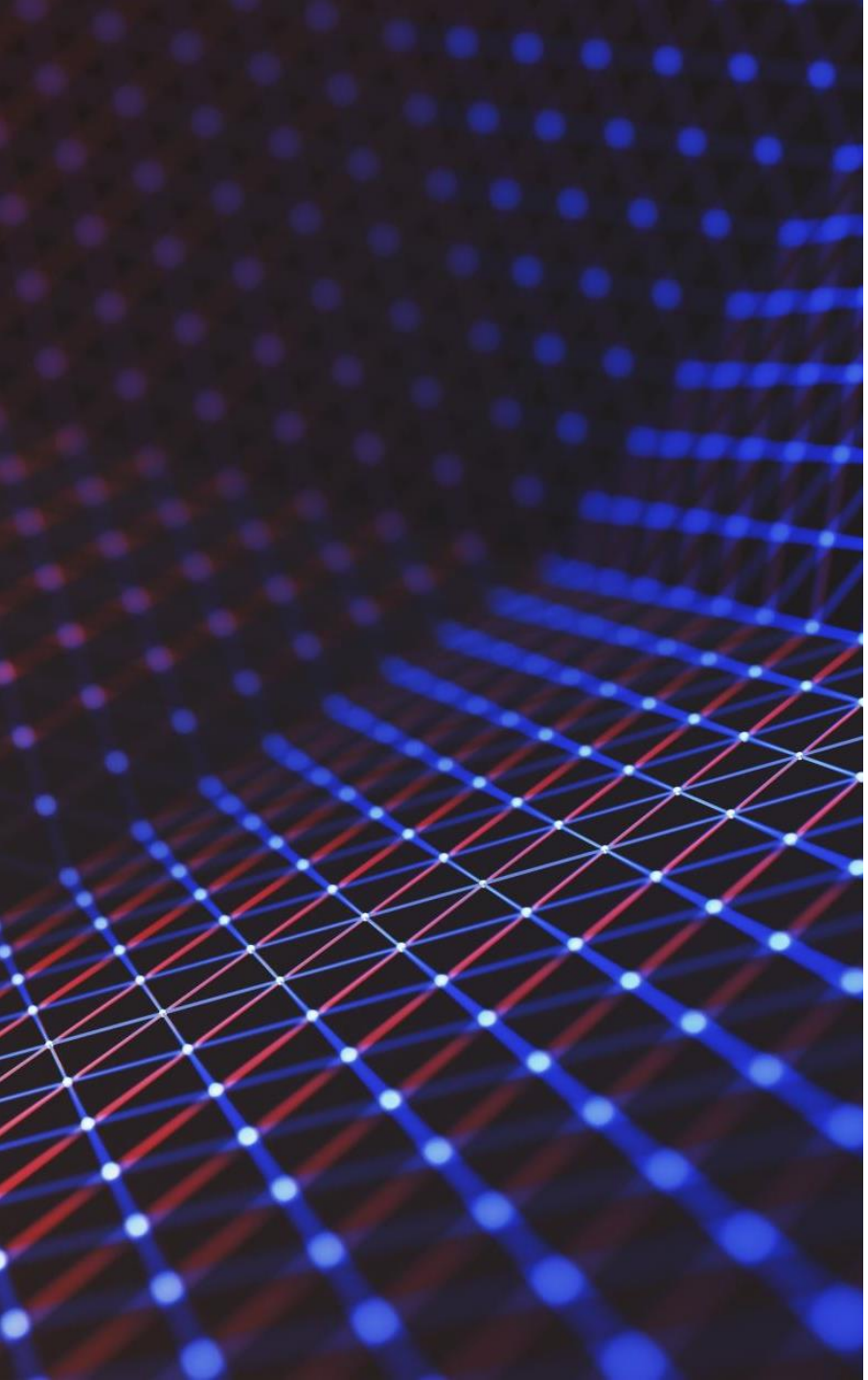
GPT-3 DECODER



- text generation

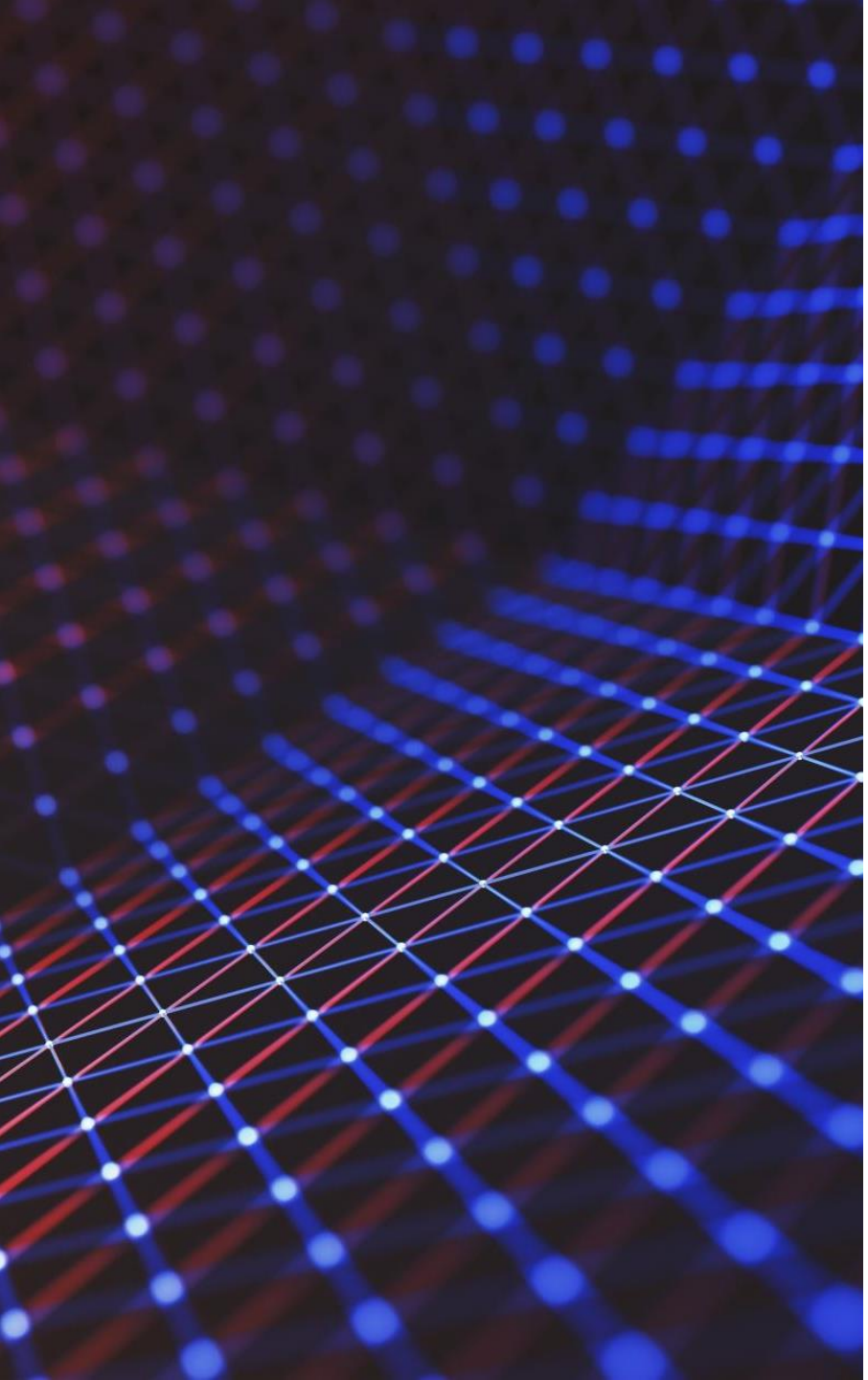
GAN (GENERATIVE ADVERSARIAL NETWORK)





LLMS (LARGE LANGUAGE MODELS)

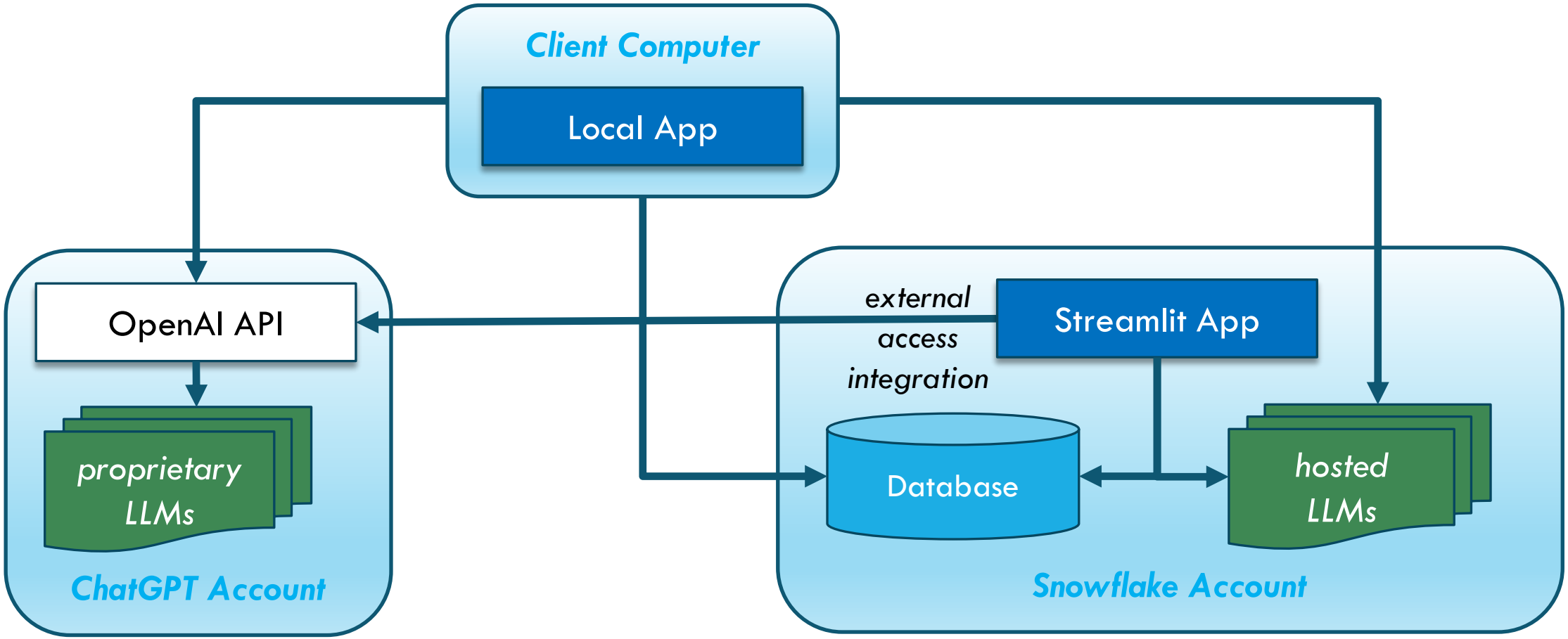
- *LLMs*
  - **Arctic** (by Snowflake) + Mistral
  - **GPT** (by OpenAI)
  - **BERT** (by Google)
  - **Bedrock** (by AWS)
  - **Claude** (by Anthropic)
- *foundational model* = LLM trained from scratch in self-supervised way (without labels)
- *fine tuning* = additional supervised training → for specific domain/task/private dataset
- *hallucination* = wrong output gen by LLM, when internal "knowledge" not for the user query



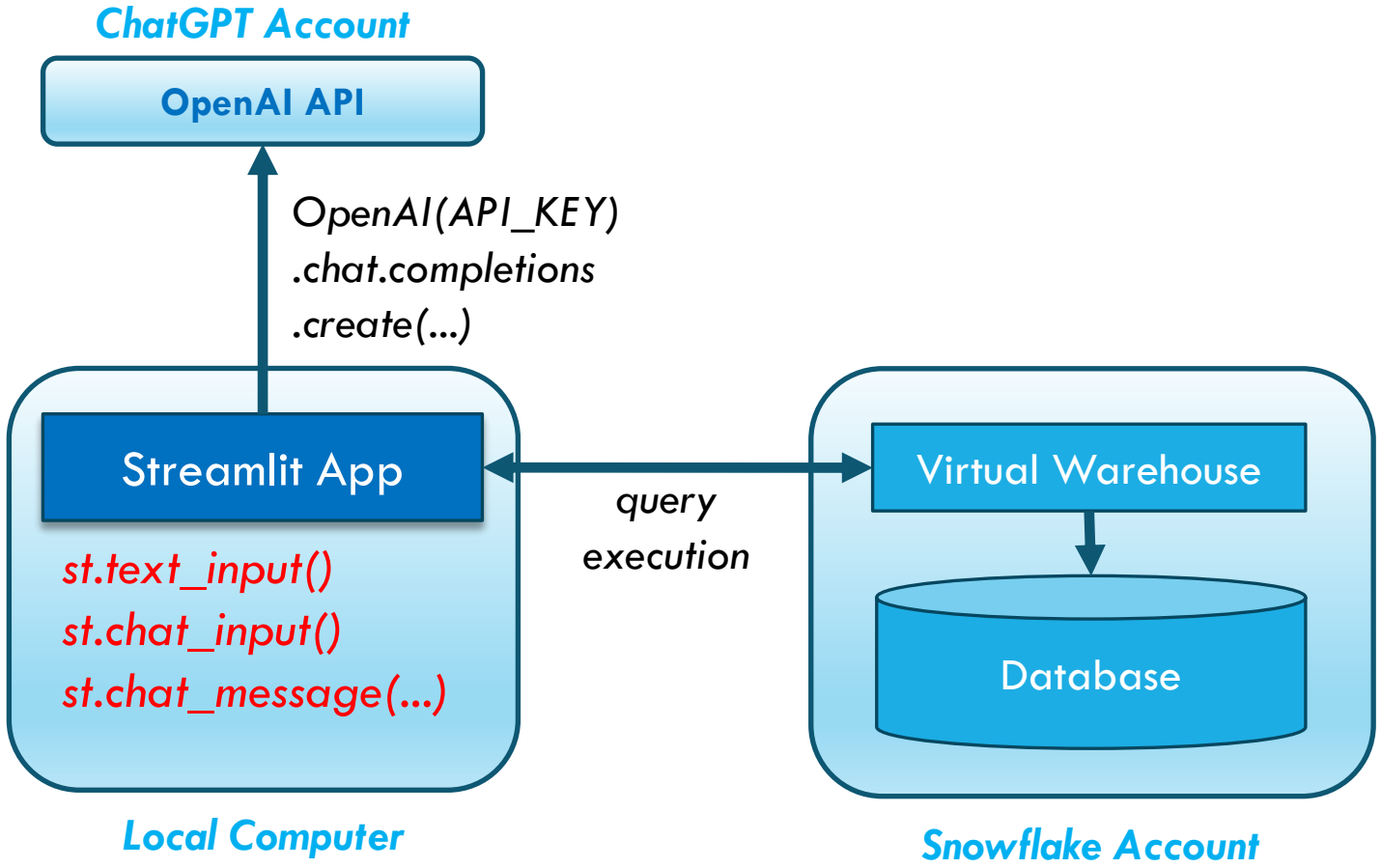
GEN-AI APPLICATIONS

- *Text-to-Text*
  - **ChatGPT** (by OpenAI)
  - **Bard** (by Google)
- *Text-to-Image*
  - **DALL-E** (by OpenAI)
  - **Midjourney**
  - **Stable Fusion** (open-source)
- *Text-to-Music*
  - **MuseGAN**

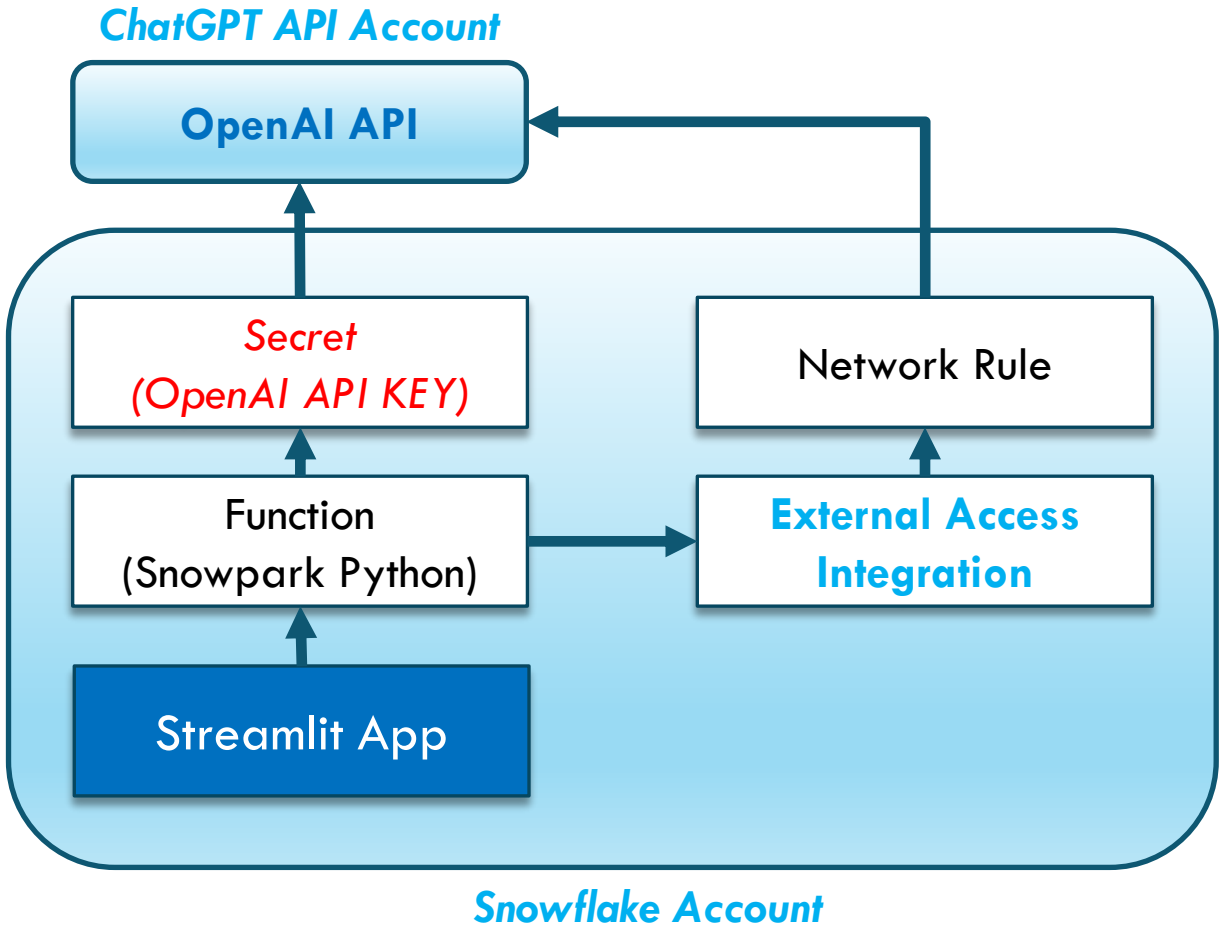
SNOWFLAKE INTEGRATIONS WITH CHATGPT

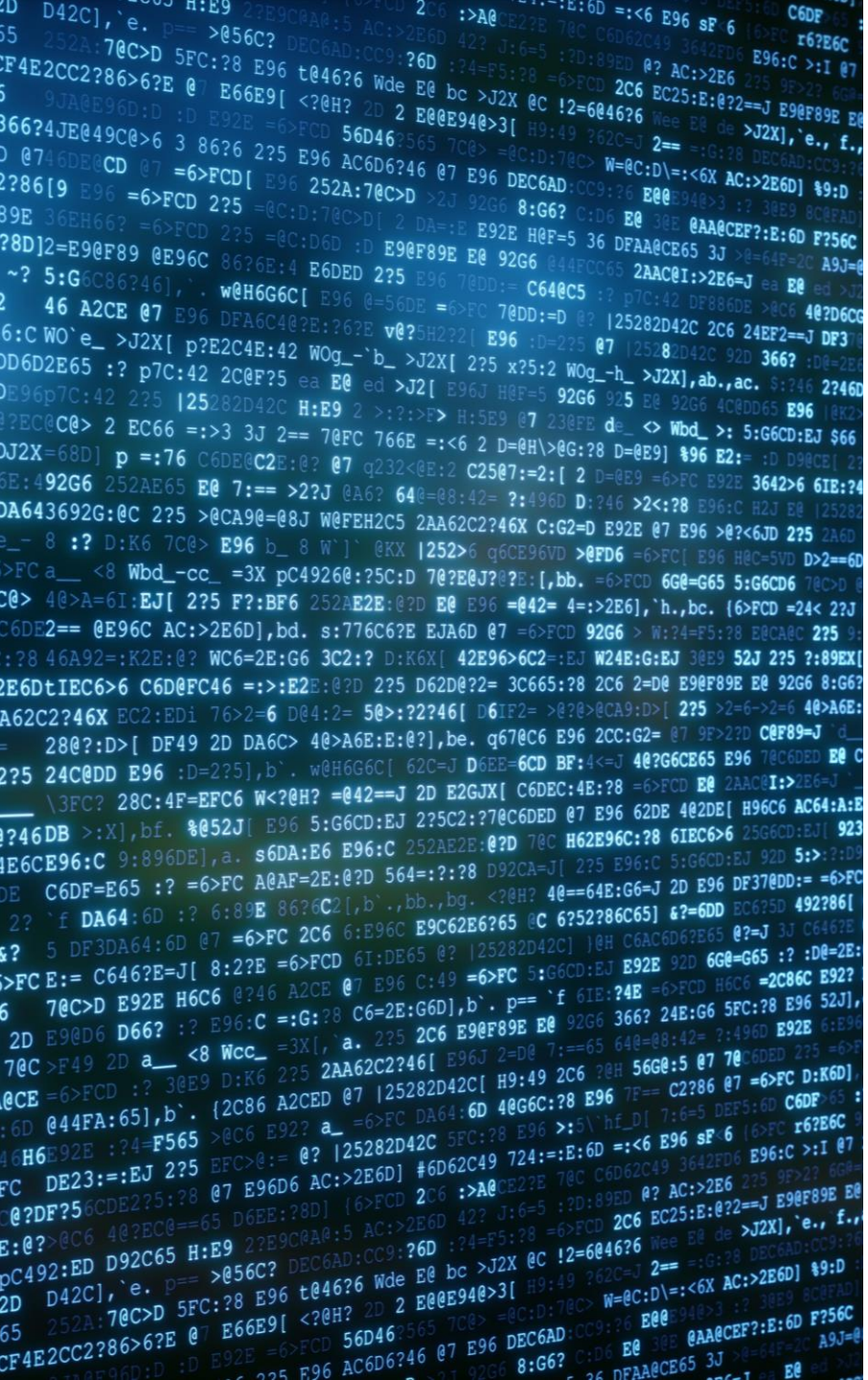


CHATGPT WITH CLIENT INTEGRATION



CHATGPT WITH EXTERNAL ACCESS INTEGRATION

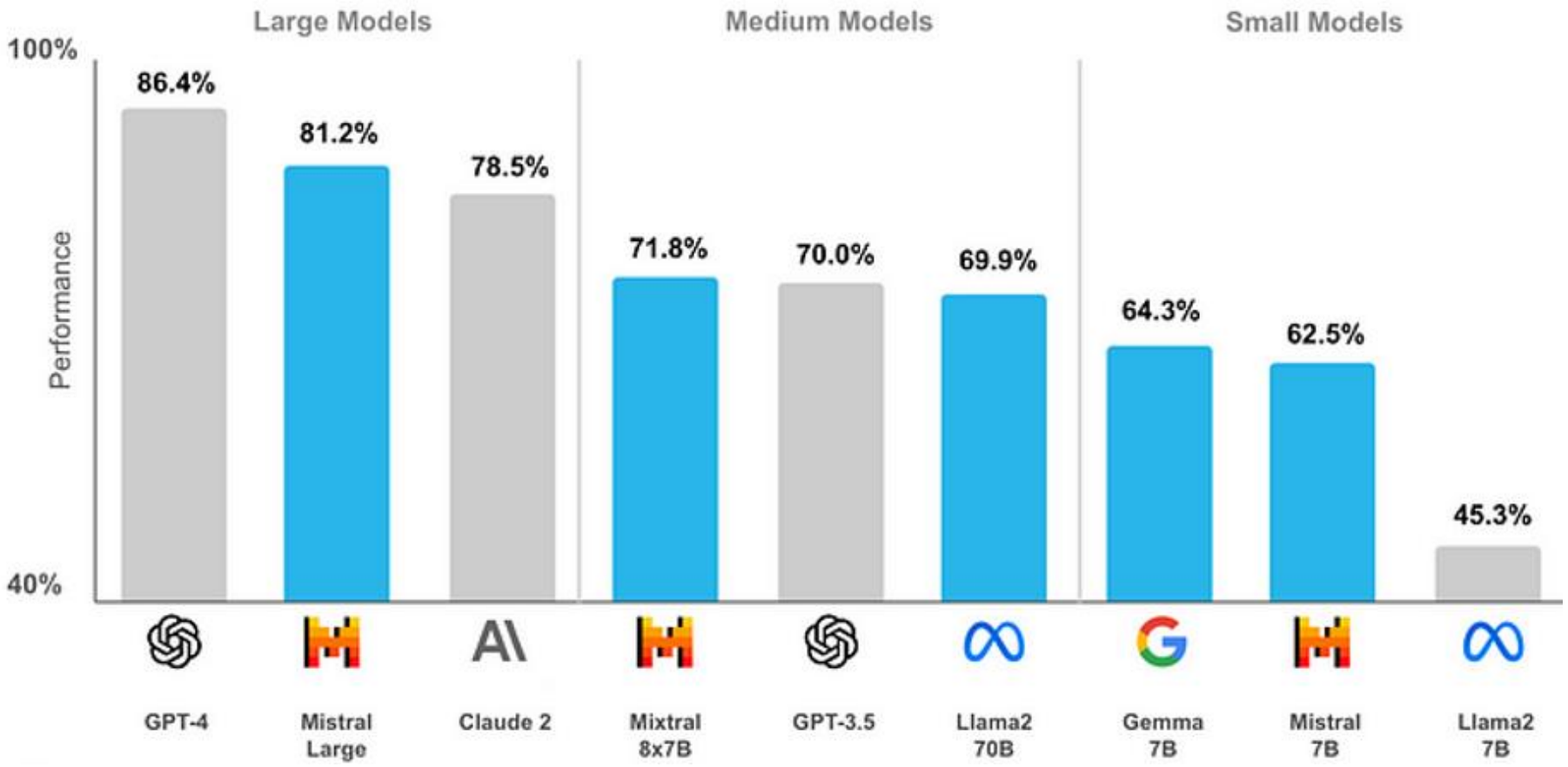


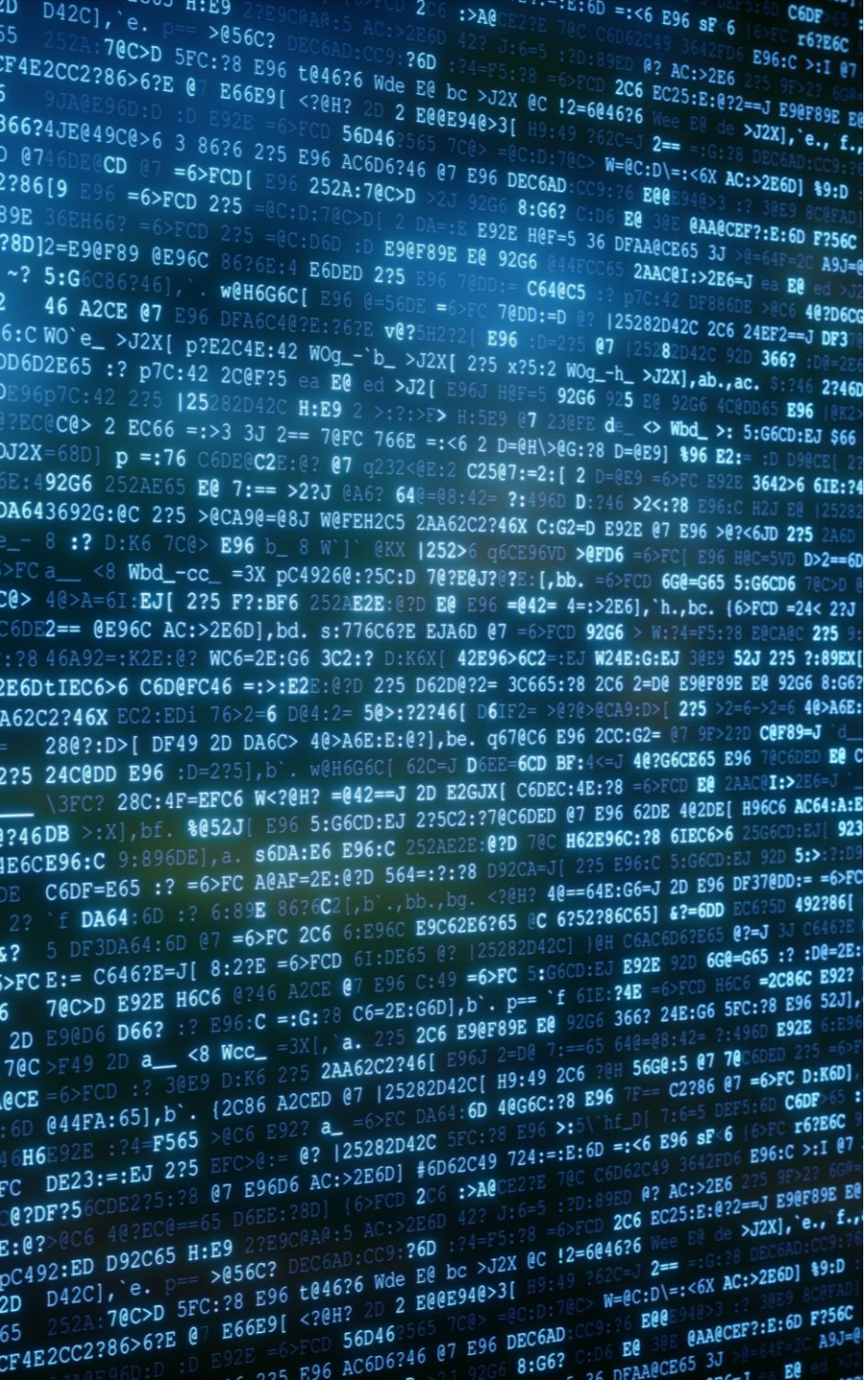


COMPLETE LLM FUNCTIONS

- = completion API, complex reasoning tasks w/ text data → choose self-hosted LLM + run requests against your data w/ custom prompts
- specialized functions (summarize, translate, sentiment) → no need to manually select an LLM or do any prompting
- stateless (does not retain context) → must pass all messages each time (bigger cost)
- ***model*** (= LLM internally stored in Snowflake)
  - 'mistral-large' – Mistral Large, by Mistral
  - 'mixtral-8x7b' – Mixtral-8x7B, from HuggingFace
  - 'llama2-70b-chat' – Llama 2, from HuggingFace, by Meta
  - 'mistral-7b' – Mistral 7B
  - 'gemma-7b' – Gemma, from HuggingFace, by Google

LLM
(LARGE LANGUAGE MODELS)

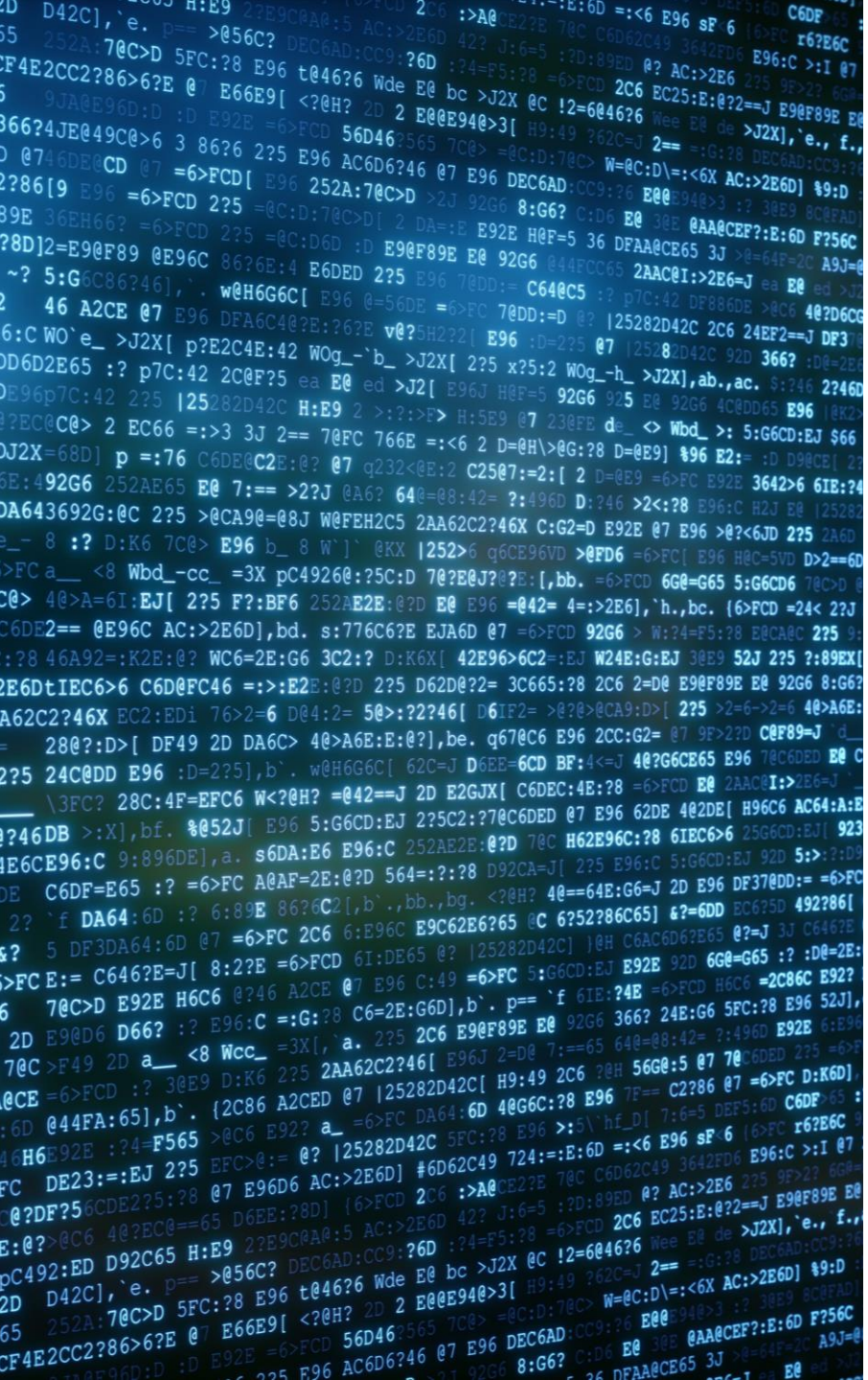




COMPLETE LLM FUNCTIONS

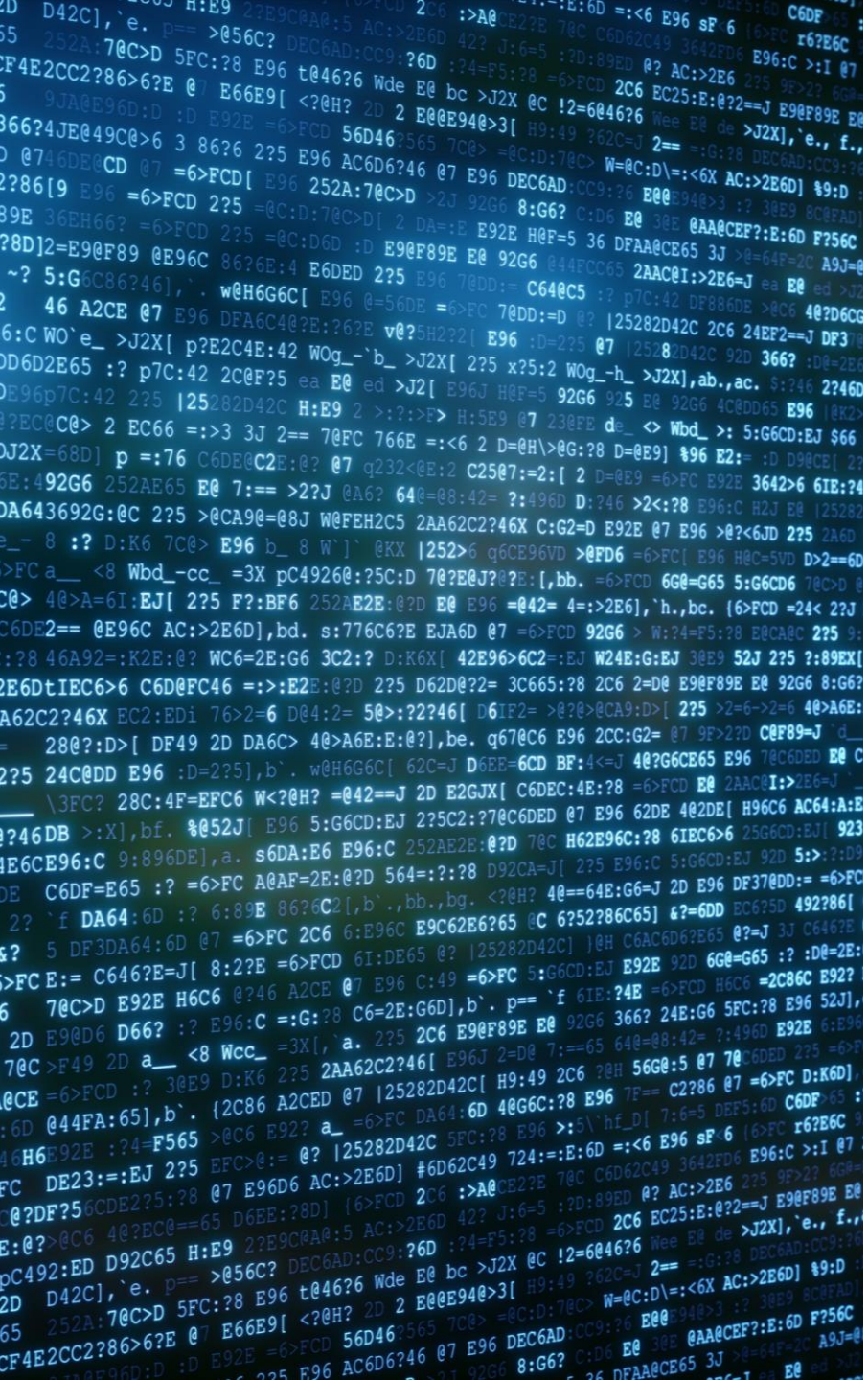
- **COMPLETE**(model varchar, messages varchar) → varchar
- **COMPLETE**(model varchar, messages array, [options object]) → object

- ***messages*** = string prompt (if first), w/ content, as ARRAY of history w/ all chat messages, w/ role/content
- ***options*** = as JSON obj, w/ temperature/top_p/max_tokens
- ***response*** = string answer (if first), w/ choices[0].messages (choices has only 1!), as JSON obj



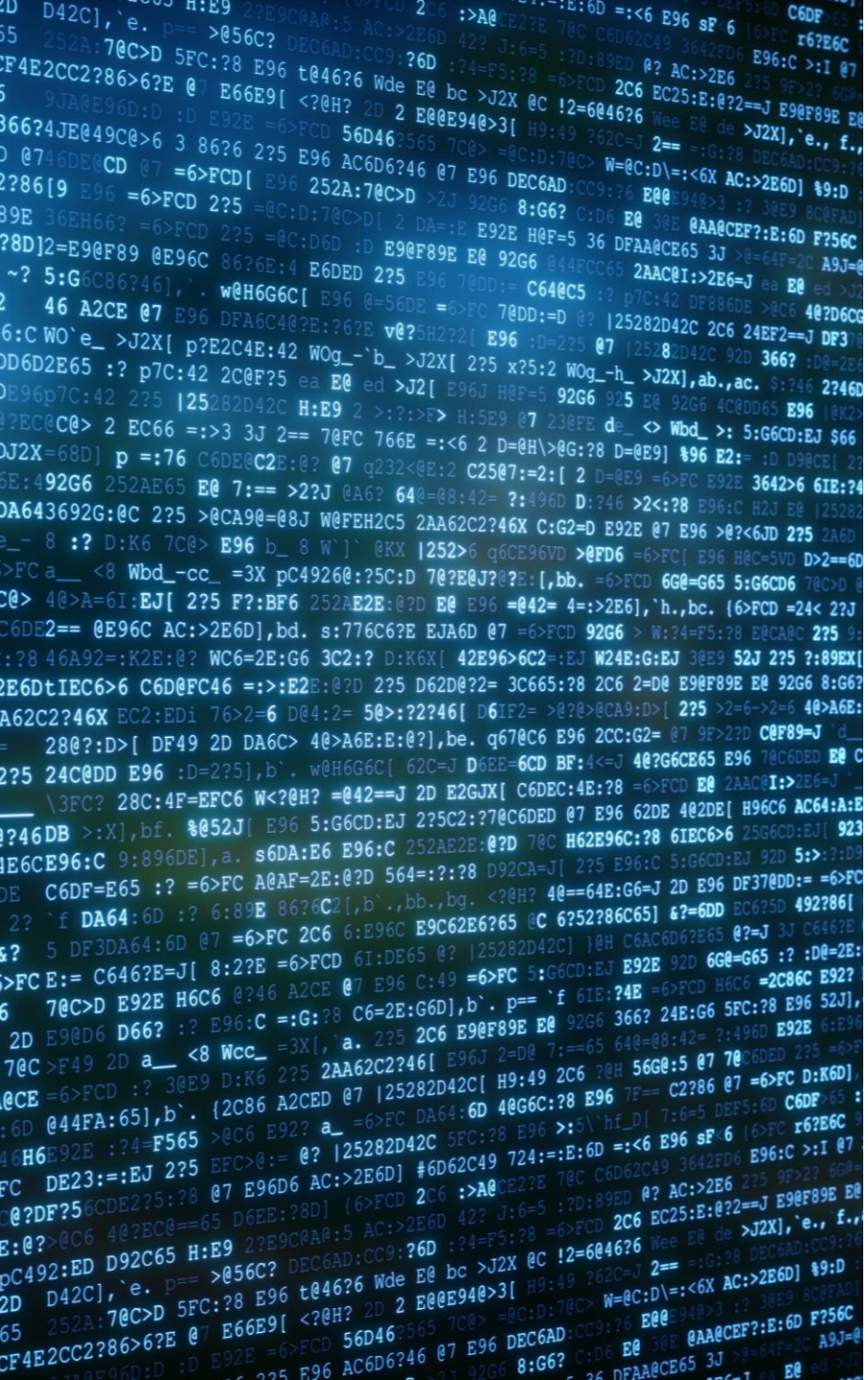
EXTRACT_ANSWER LLM FUNCTION

- **EXTRACT_ANSWER**(from_text varchar, question varchar) → array (as JSON obj w/ answer + score)
- = extract info from unstructured/semi-structured data → text doc or English JSON obj
- this doesn't formulate an answer in NL!



SENTIMENT LLM FUNCTION

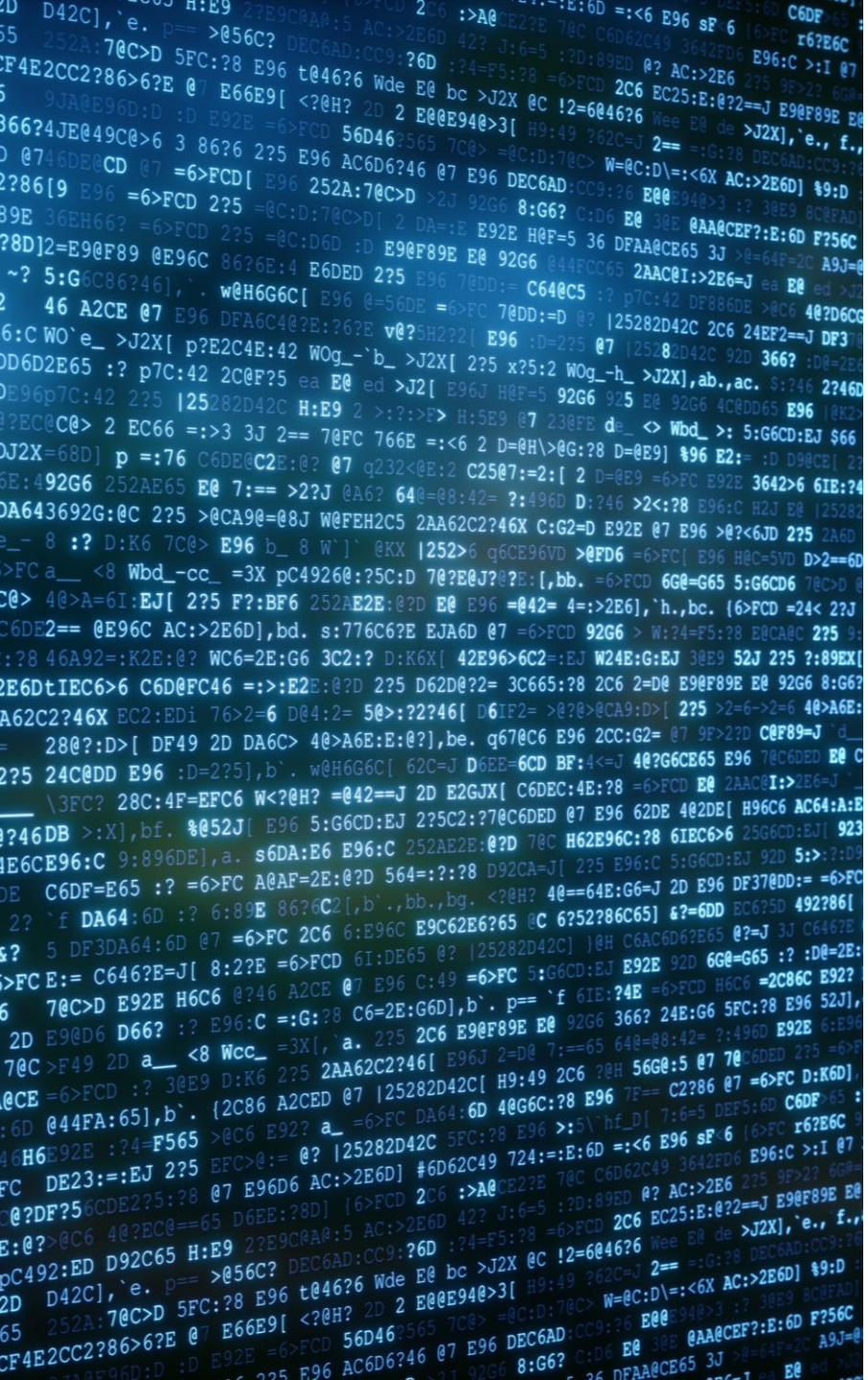
- **SENTIMENT**(text varchar) → float [-1 .. 1], 0 for neutral, for English only
- ***sentiment analysis*** = NLP problem, to extract subjective info (like sarcasm) as polarity (pos/neg/neutral), usually using reviews, detect mood/tone of text
- binary sentiment classifier → Good/Bad (mood) + sarcasm
- w/ TensorFlow, w/ RNN
- on IMDB reviews
- encoding words/terms as numeric → generate/visualize vectors



SUMMARIZE LLM FUNCTION

- **SUMMARIZE**(text varchar, [options object]) → varchar

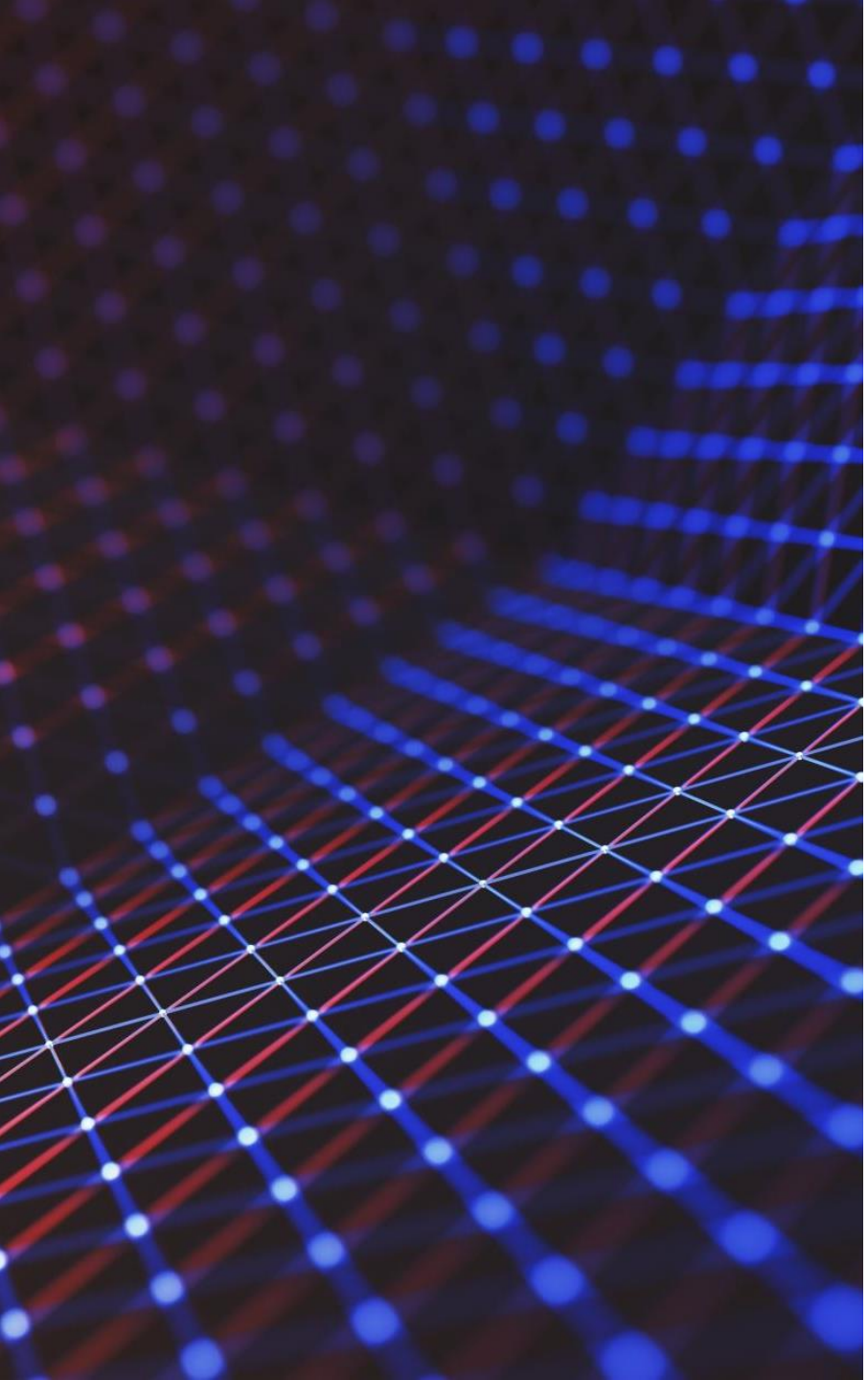
- = automatically get a synopsis of text
- = summarizes the given English-language input text.



TRANSLATE LLM FUNCTION

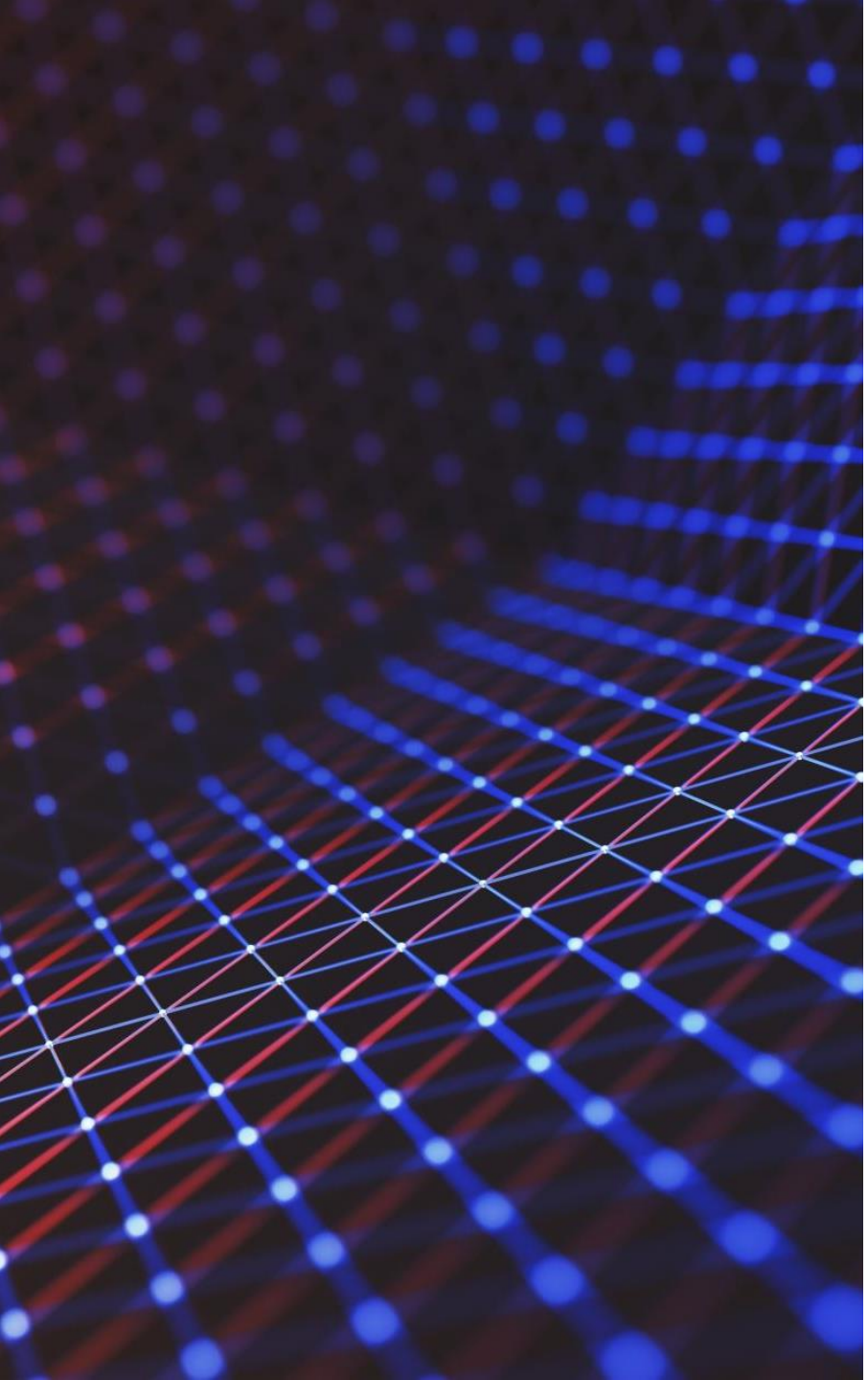
- **TRANSLATE**(text varchar, from_lang varchar, to_lang varchar) → varchar

- = translate docs to other languages
- *langs*: en, fr, ge, it, ja, ko, pl, pt, ru, es, sv
- pass empty lang string ‘’ to auto-detect



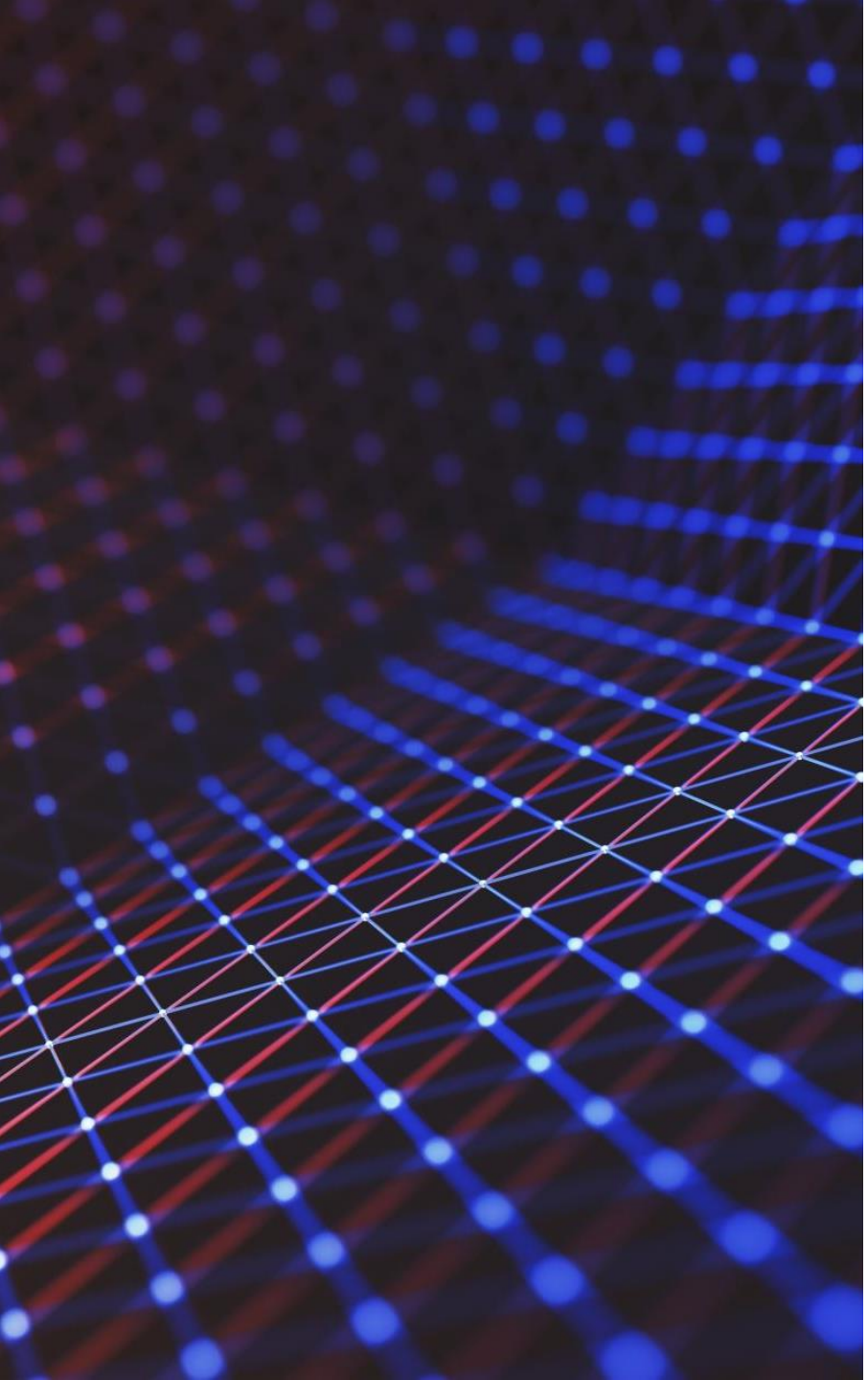
ACCESS RIGHTS TO LLM FUNCTIONS

- **SNOWFLAKE.CORTEX_USER** database role
- assigned initially only to ACCOUNTADMIN role
- GRANT DATABASE ROLE CORTEX_USER → TO ROLE *some account role* → TO USER *some user*
- grant to PUBLIC role → to make available to everyone
- no granular access → can access ***all*** LLM functions or none
- also available in trial accounts (to ACCOUNTADMIN)



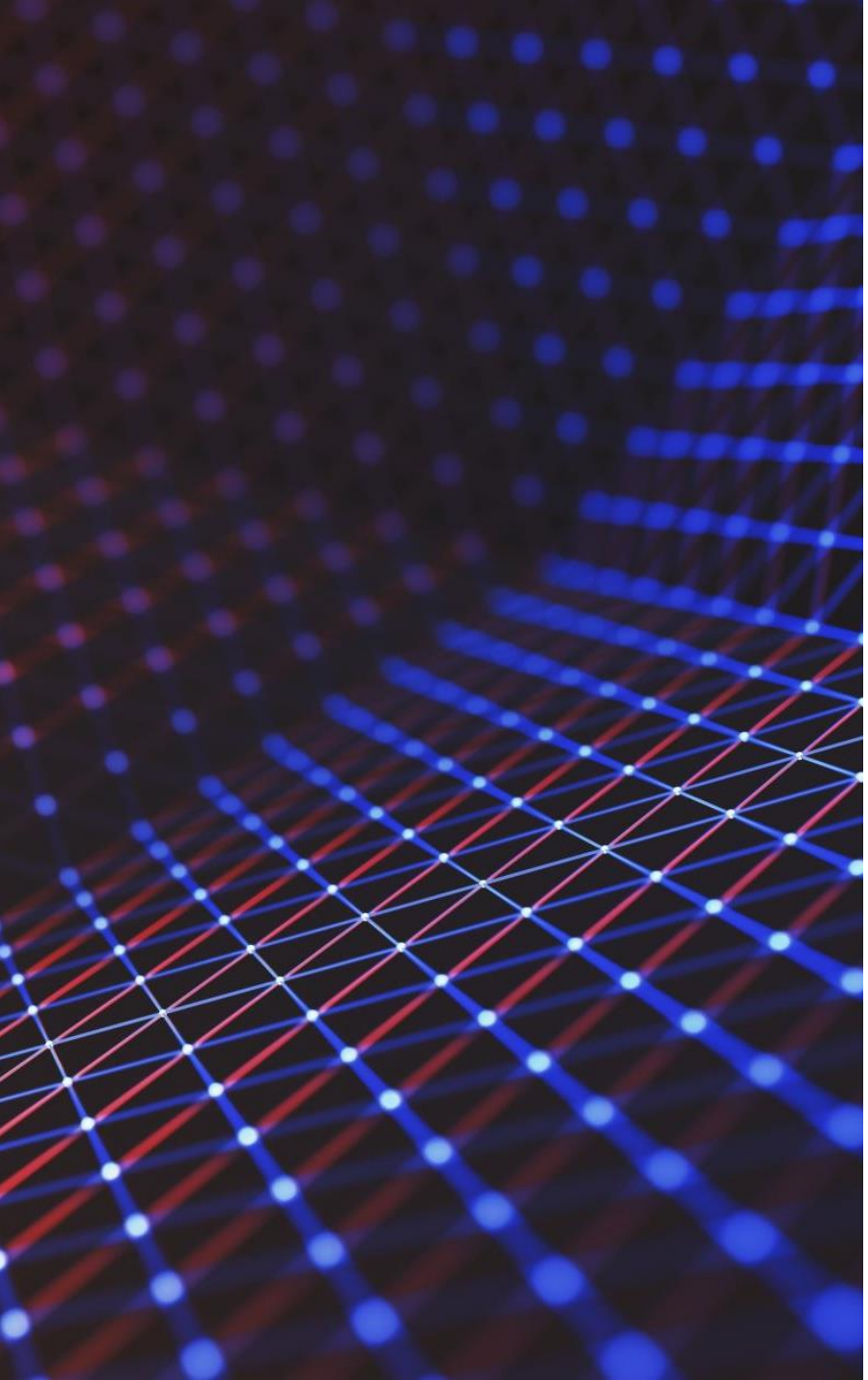
COST OF LLM FUNCTIONS (AND UI EXTENSIONS)

- *charged based on*
  - 1M I/O ***tokens*** (1 token ~= 4 words)
  - ***quotas*** = TPM (tokens/min) + RPM (rows/min) → throttling
  - ***context window*** = max input tokens per call
- warehouse compute credits → use max MEDIUM (larger WHs do not improve LLM performance + could trigger throttling)
- max 1 credit/day for free trial accounts



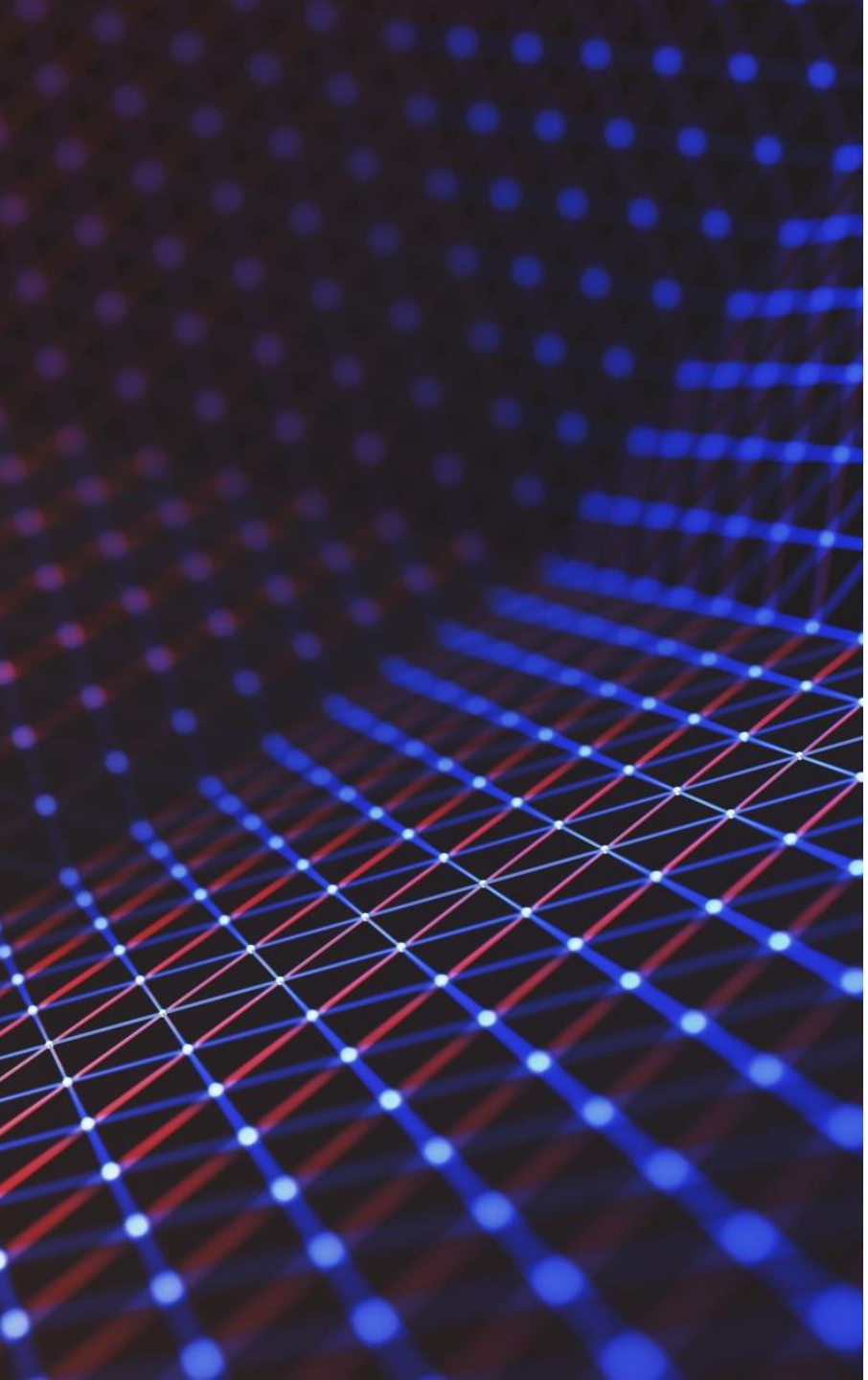
COMPUTE COSTS + QUOTAS + CTX WINDOW PER LLM FUNCTION

- **Translate** → 0.33 credits/1M tokens, 1M TPM, 2K RPM, 1K ctx window
- **Summarize** → 0.10 credits/1M tokens, 300K TPM, 500 RPM, 52K ctx window
- **Extract Answer** → 0.08 credits/1M tokens, 1M TPM, 3K RPM, 2K text + 64 question ctx window
- **Sentiment** → 0.08 credits/1M tokens, 1M TPM, 5K RPM, 512 ctx window
- **Embed Text** → 0.03 credits/1M tokens



COMPUTE COSTS + QUOTAS + CTX WINDOW PER COMPLETE MODEL

- **mistral-large** → **5.10 credits/1M tokens**, 200K TPM, 100 RPM, 32K ctx window
- **llama2-chat-70b** → 0.45 credits/1M tokens, 300K TPM, 400 RPM, 4K ctx window
- **reka-flash** → 0.45 credits/1M tokens, 300K TPM, 400 RPM, 8K ctx window
- **mixtral-8x7b** → 0.22 credits/1M tokens, 300K TPM, 400 RPM, 32K ctx window
- **mistral-7b** → 0.12 credits/1M tokens, 300K TPM, 500 RPM, 32K ctx window
- **gemma-7b** → 0.12 credits/1M tokens, 300K TPM, 500 RPM, 8K ctx window



COMPLETE MODELS

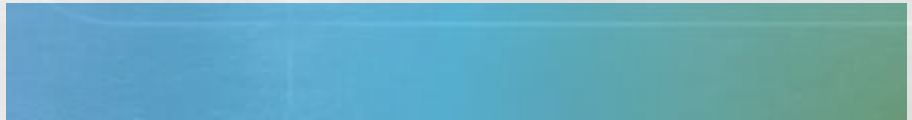
- **mistral-large** → for synthetic text/code gen, agents
- **reka-flash** → to write product descriptions or blog posts + coding
- **mixtral-8x7b** → for text generation, classification, question answering
- **llama2-70b-chat** → for extracting data, help write job descriptions
- **mistral-7b** → for simple summarization, structuration, quick question answering tasks
- **gemma-7b** → for simple code/text completion tasks

LLM UI EXTENSIONS

Snowflake Copilot

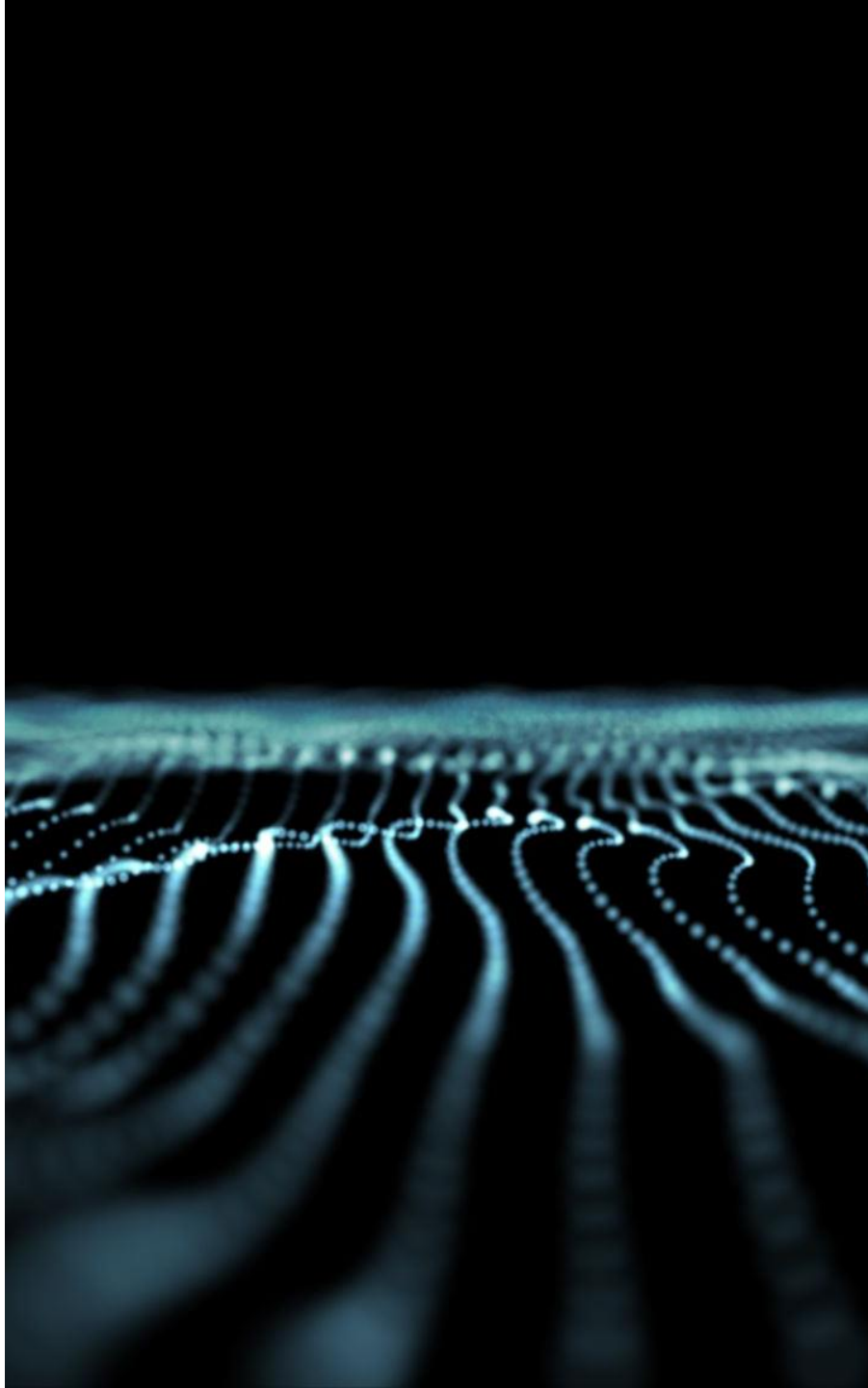
Universal Search

Document AI

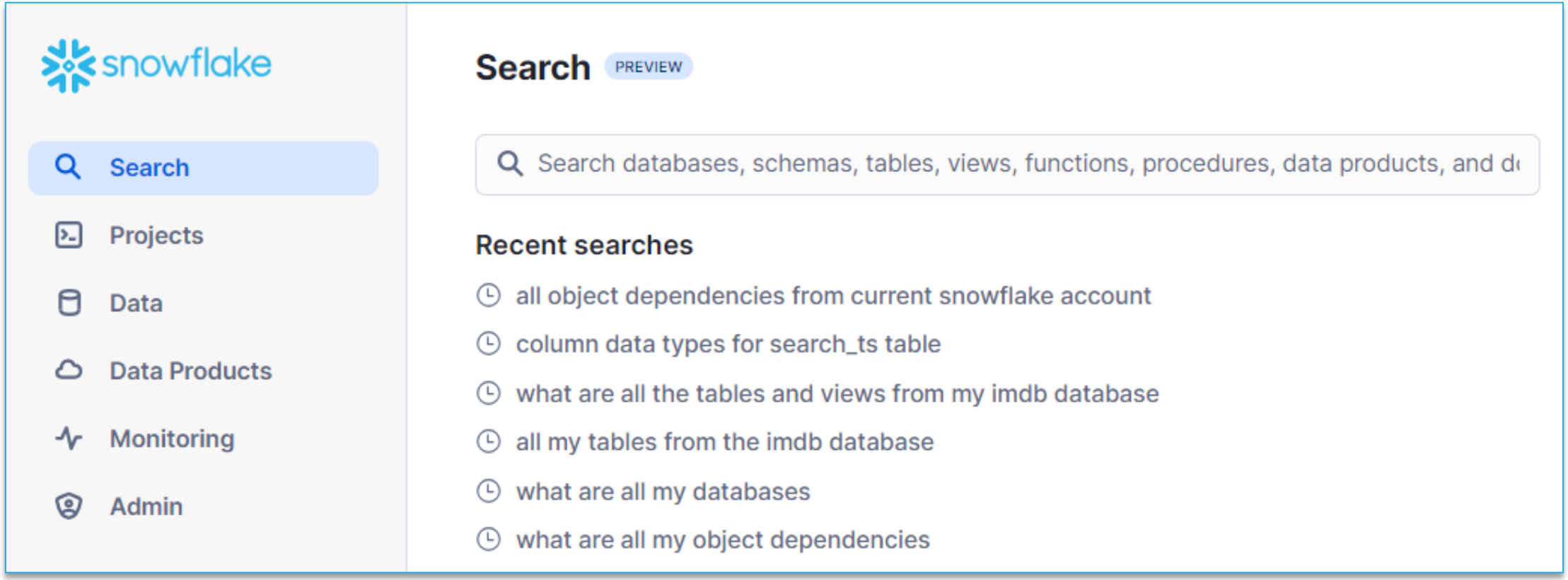


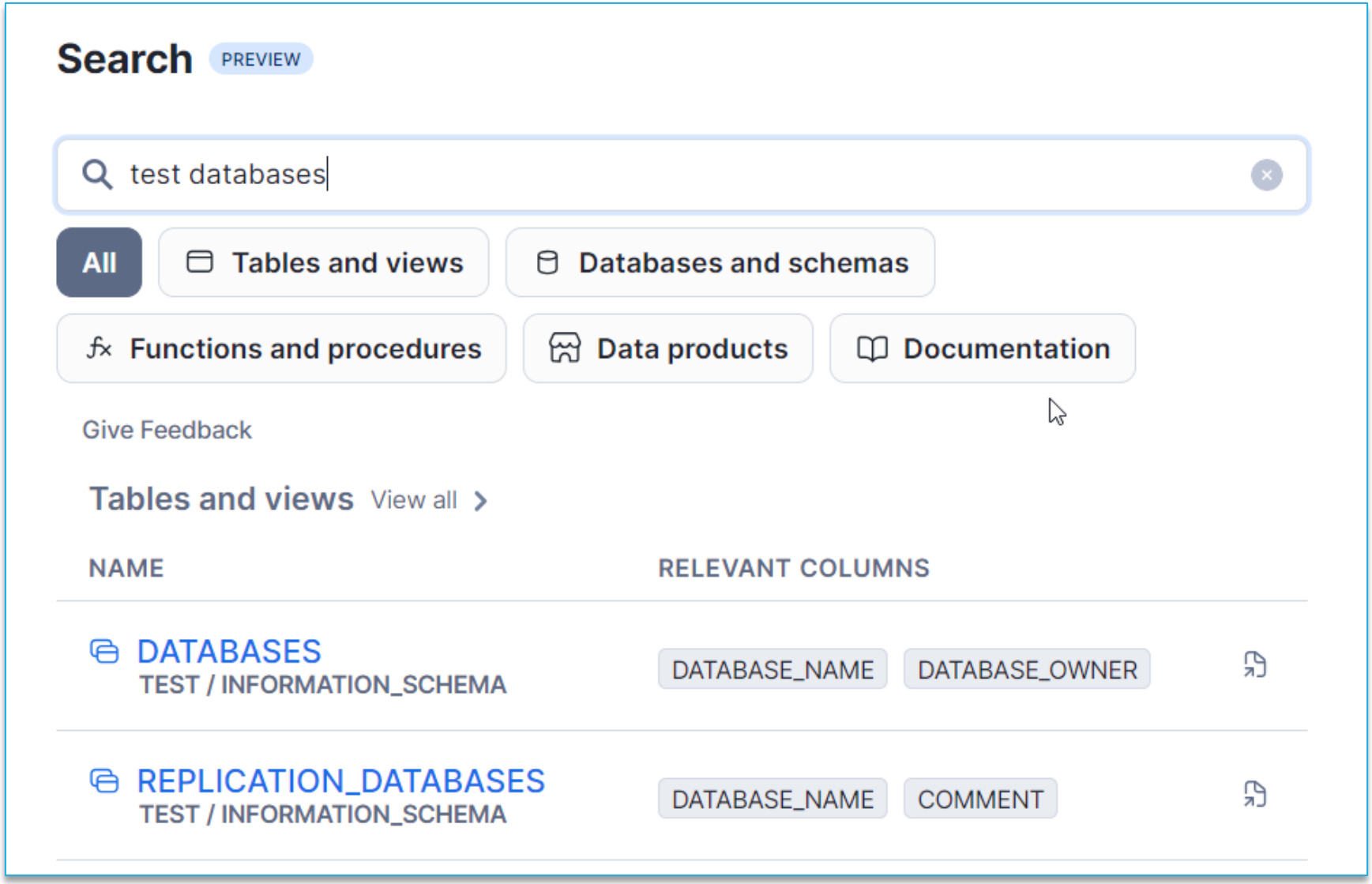
LLM SNOWSIGHT UI EXTENSIONS

- **Universal Search**
  - own db objs: dbs/schemas, tables/views, functions/procs
  - Snowflake Marketplace data products
  - Snowflake Docs + Community KB
- **Snowflake Copilot**
  - gen SQL queries from NL → ***Text2SQL*** (programatically)
  - uses Universal Search for metadata
  - (parallel to LangChain for ChatGPT)
- **Document AI**
  - search content in own unstructured data/docs
  - access to private data (metadata for Univ. Search!)
  - (parallel to LlamaIndex for ChatGPT)



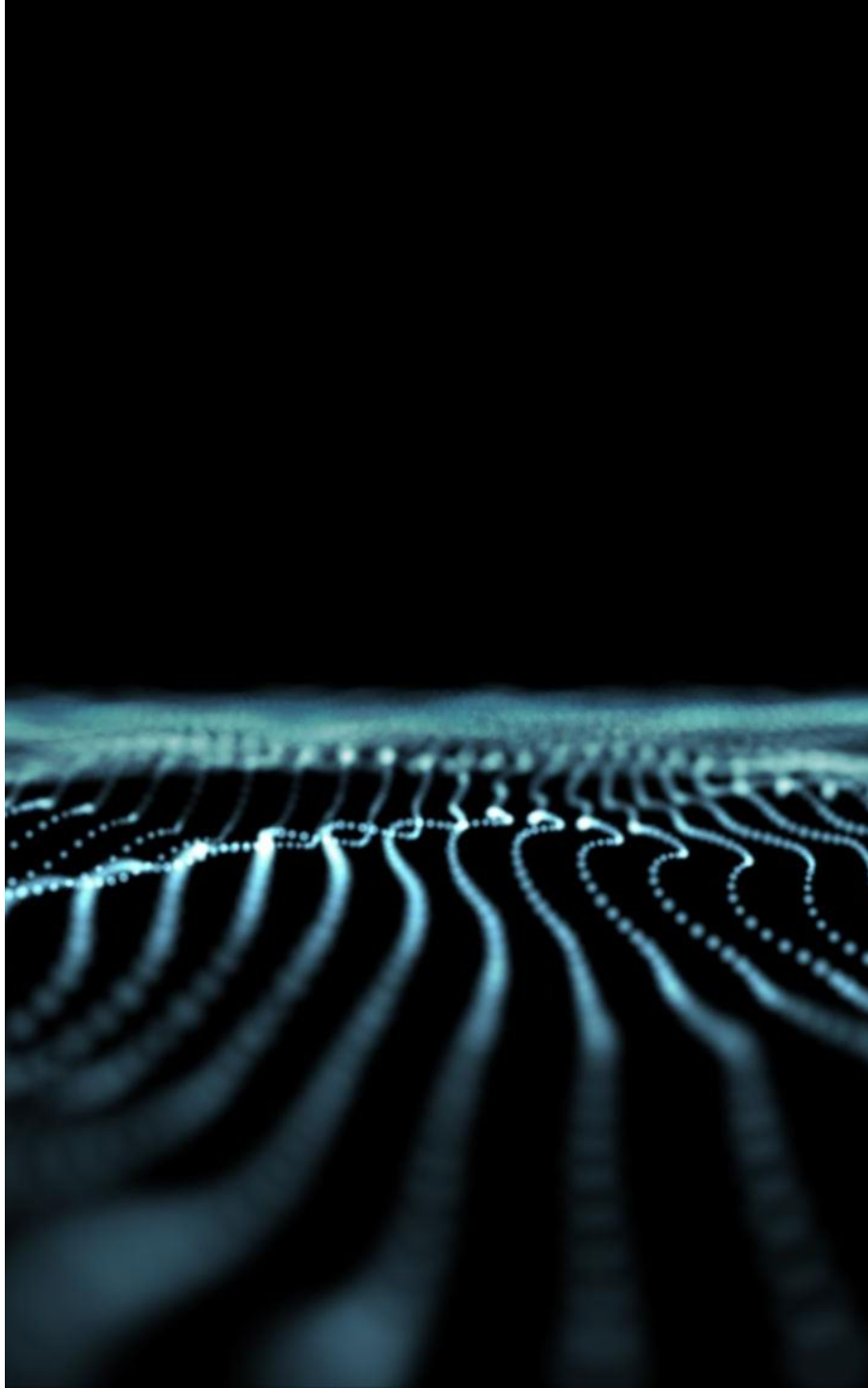
UNIVERSAL SEARCH IN SNOWSIGHT

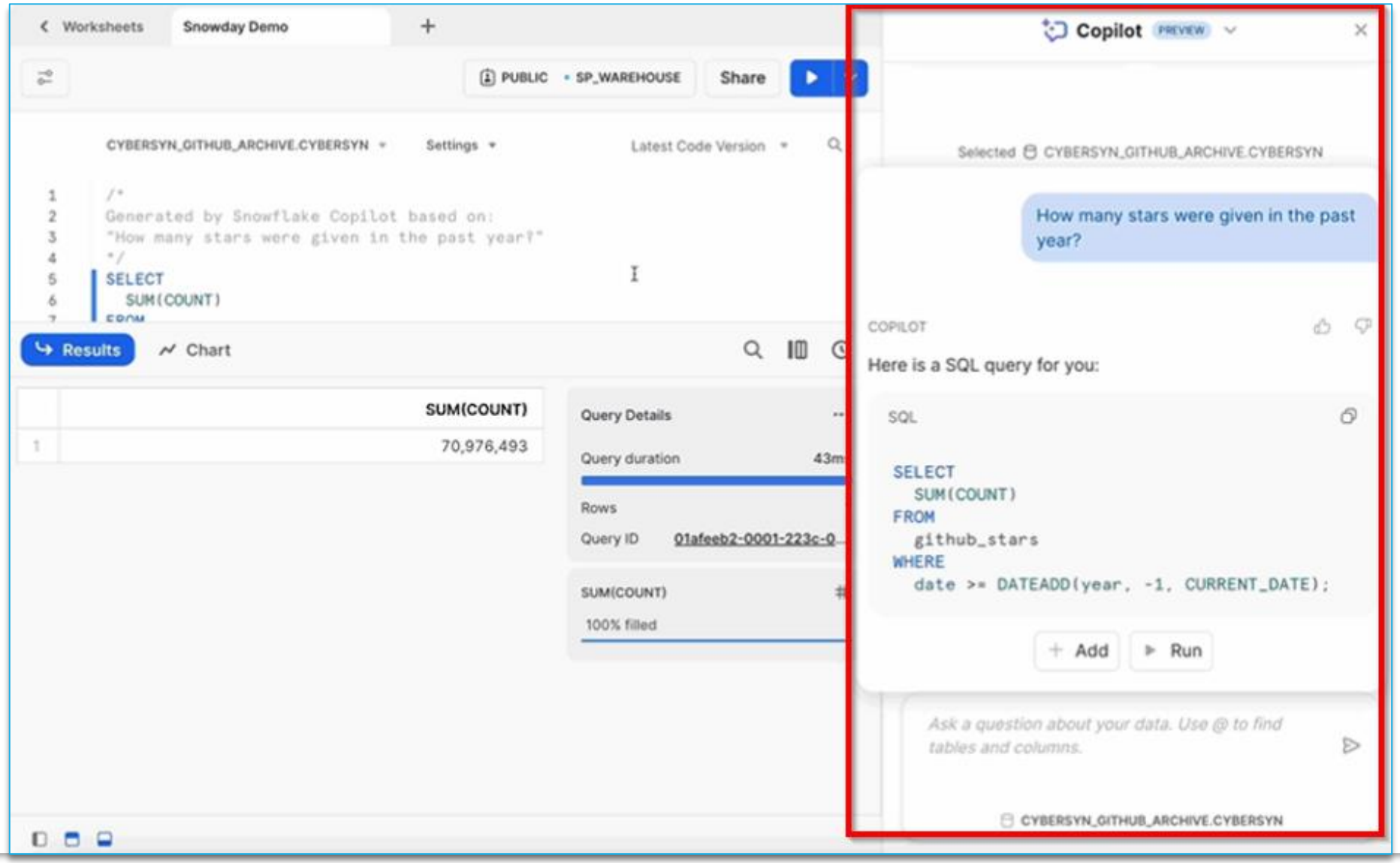



Search
PREVIEW
test databases
All
Tables and views
Databases and schemas
Functions and procedures
Data products
Documentation
Give Feedback
Tables and views
View all
NAME
RELEVANT COLUMNS
DATABASES
TEST / INFORMATION_SCHEMA
DATABASE_NAME
DATABASE_OWNER
REPLICATION_DATABASES
TEST / INFORMATION_SCHEMA
DATABASE_NAME
COMMENT

UNIVERSAL SEARCH

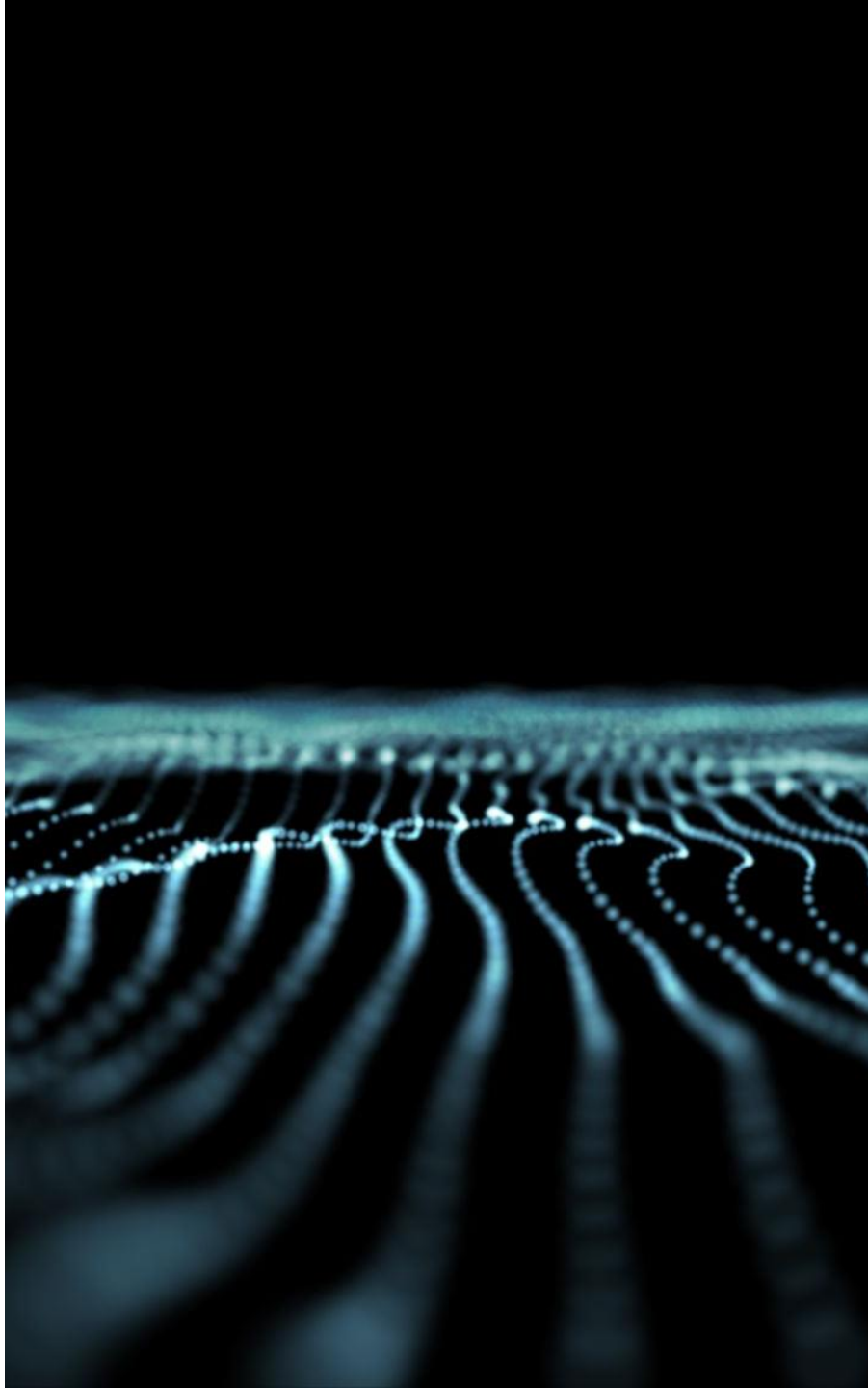
- **Universal Search** = NL ML/LLM-powered search (from Neeva) to quickly discover/access metadata/apps
- in PuPr since Mar 2024 (in some AWS/Azure regs) → new Search tab and field on top of Snowsight
- find own account dbs / schemas / tables&views / functions&procs / cols → w/ latency (~hours)
- search data prods (datasets & native apps) from the Snowflake Marketplace
- search Snowflake Doc topics (+ Other Docs) & Snowflake Community KB articles
- auto-used by Snowflake Copilot for table & column names, to generate SQL queries

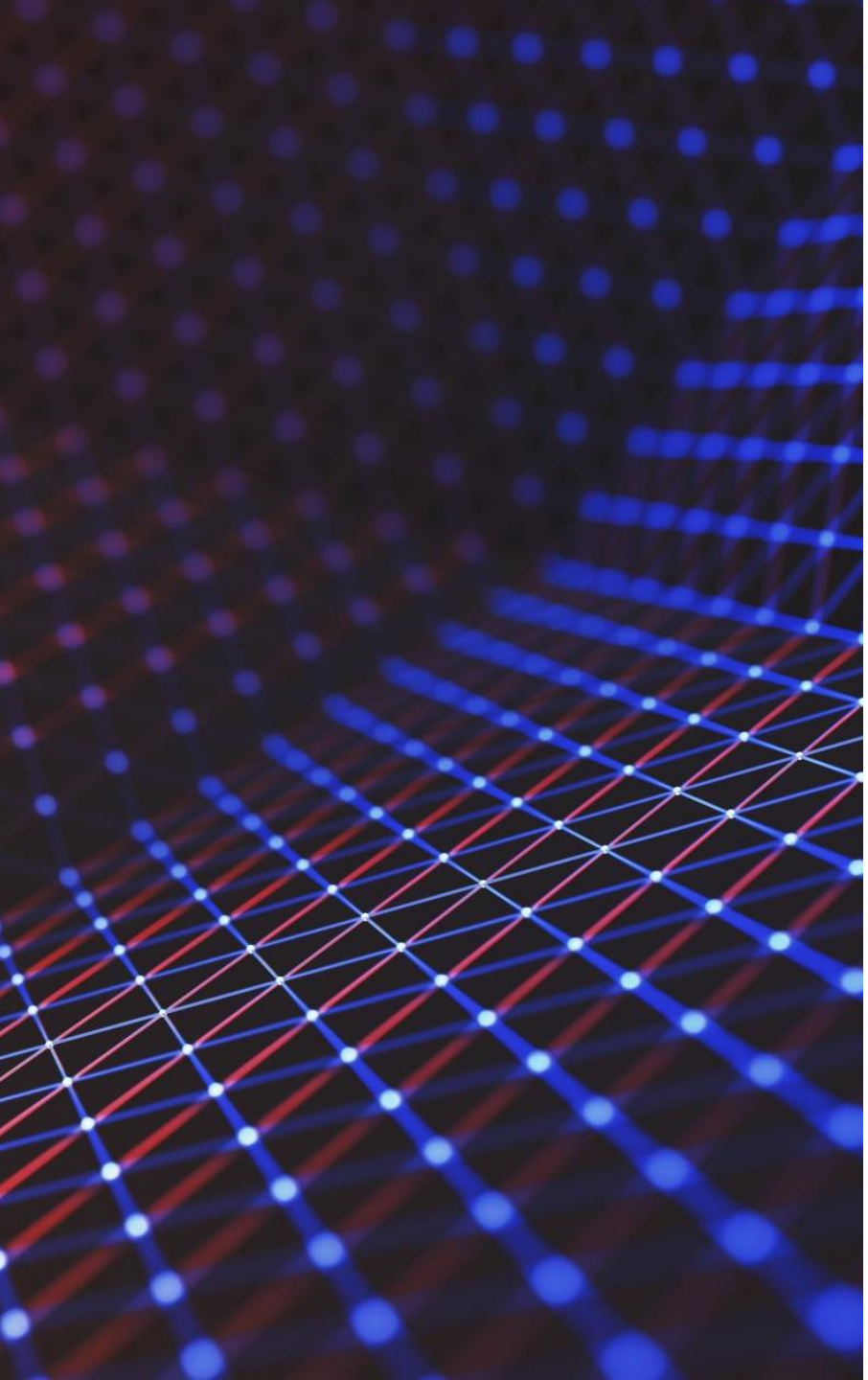



Worksheets
Snowday Demo
PUBLIC
SP_WAREHOUSE
Share
CYBERSYN_GITHUB_ARCHIVE.CYBERSYN
Settings
Latest Code Version
/*
Generated by Snowflake Copilot based on:
"How many stars were given in the past year?"
*/
SELECT
SUM(COUNT)
Results
Chart
SUM(COUNT)
1
70,976,493
Query Details
Query duration
43m
Rows
Query ID
01afeeb2-0001-223c-0
SUM(COUNT)
100% filled
Copilot
PREVIEW
Selected CYBERSYN_GITHUB_ARCHIVE.CYBERSYN
How many stars were given in the past year?
COPILOT
Here is a SQL query for you:
SQL
SELECT
SUM(COUNT)
FROM
github_stars
WHERE
date >= DATEADD(year, -1, CURRENT_DATE);
Add
Run
Ask a question about your data. Use @ to find tables and columns.
CYBERSYN_GITHUB_ARCHIVE.CYBERSYN

SNOWFLAKE COPILOT

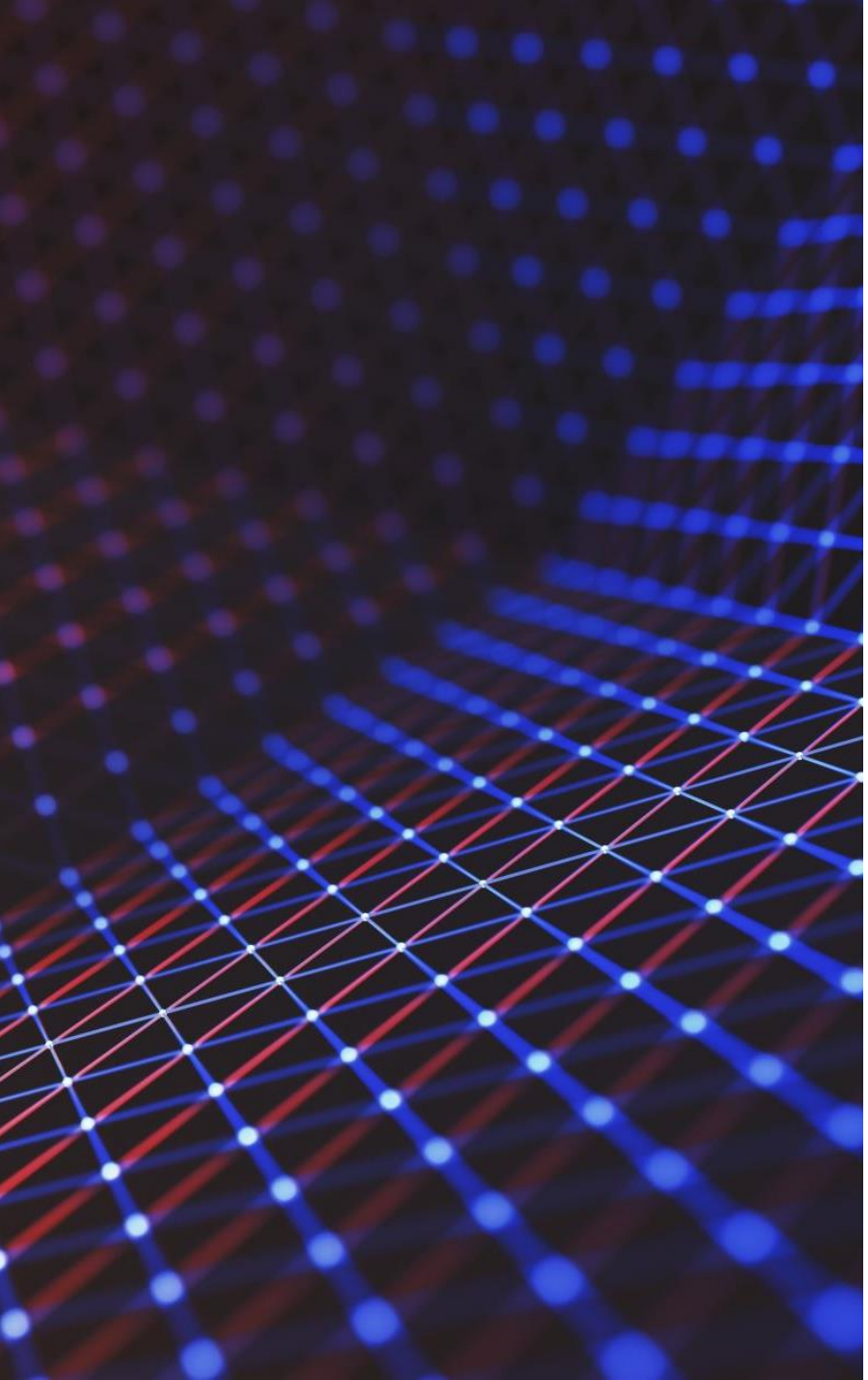
- **Snowflake Copilot** = LLM-powered assistant (Llama2 fine-tuned by Snowflake) to generate and refine SQL for your current dbs with NL (metadata access only!)
- in PuPr (Apr 2024, in selected AWS regions), will be charged separately!
- uses Universal Search for access to metadata
- **Text2SQL** = generate SQL from NL prompts programmatically (w/ the same LLMs behind Snowflake Copilot)
- ~LangChain for ChatGPT → generate SQL for own databases & tables





SNOWFLAKE COPILOT QUERY GENERATION

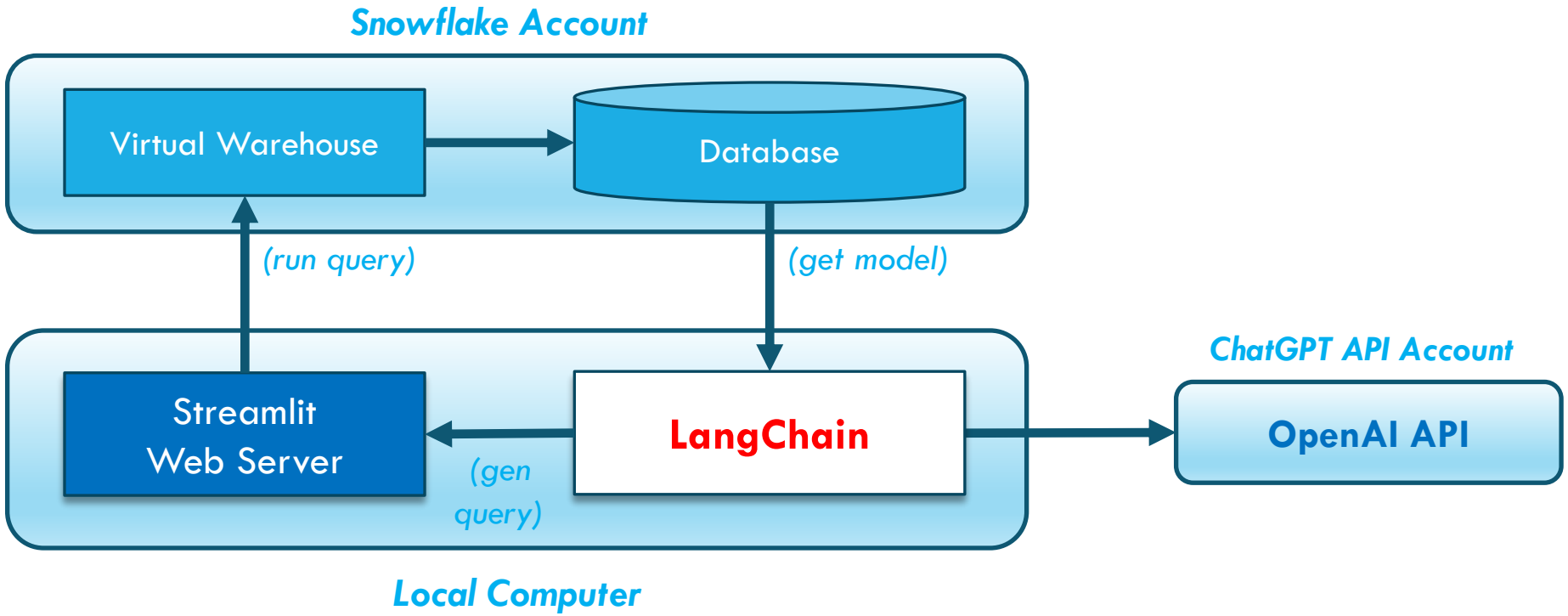
- in one single SQL worksheet, w/ Run/Add query buttons
- in one selected db+schema (use views for cross-joins)
- full data analysis/exploration (in English only)
- only metadata is accessed and queried (RBAC respected)
- all your data remains secure in Snowflake
- must provide data values if needed
- @table/column name prefix + only top 10 considered
- learn how data is structured + get exploring suggestions
- ~3-4h to detect new dbs/schema/tables

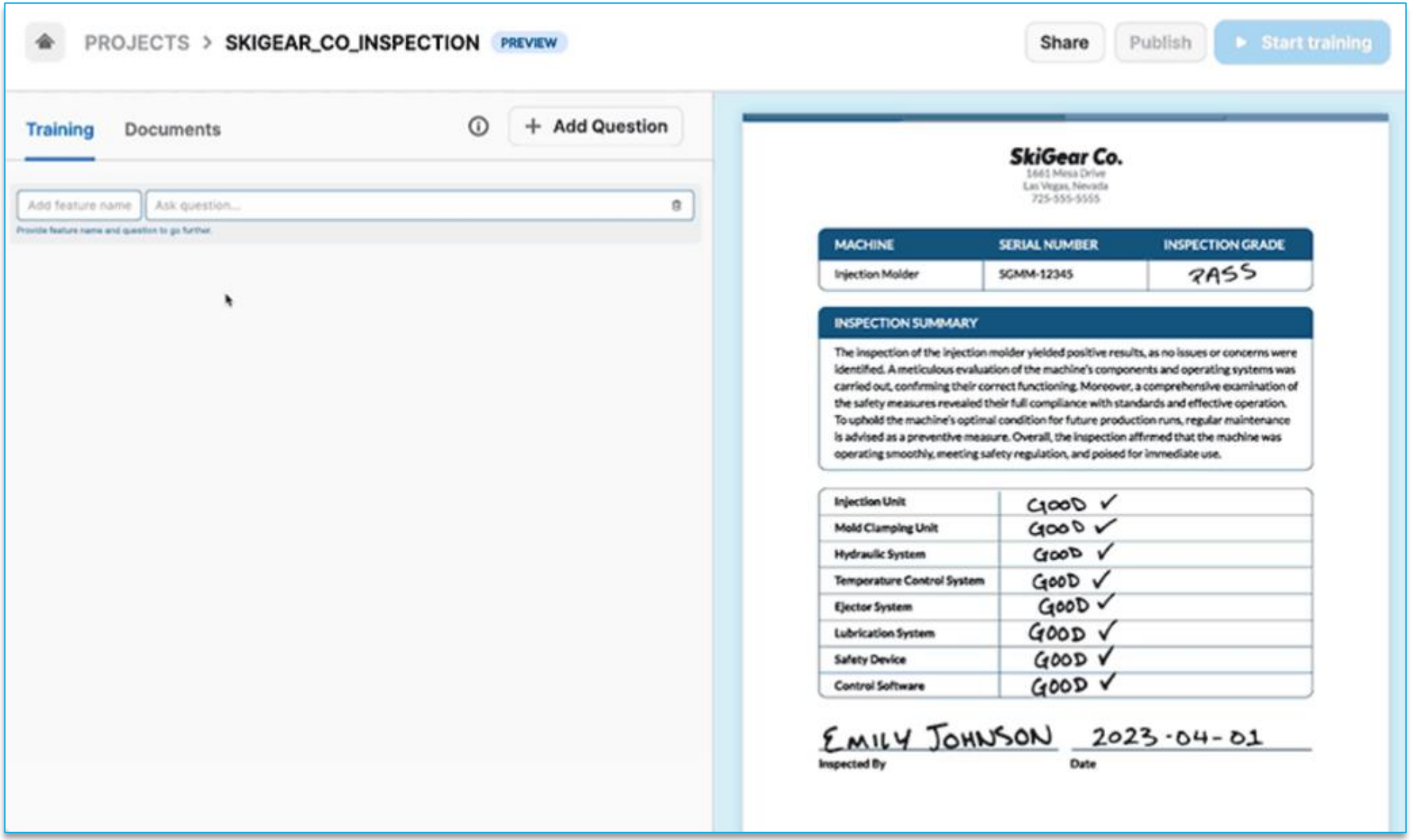


SNOWFLAKE COPILOT QUERY TUNING

- guided conversation, w/ thumbs up/down buttons
- explain step-by-step what a query does
- optimize existing query, through recommendations
- suggest query fixes
- improve query fluency + help create curated views
- allows follow-up questions to build complex queries
- can add specific Snowflake features
- helps you learn Snowflake and SQL
- explain Snowflake features, based on doc

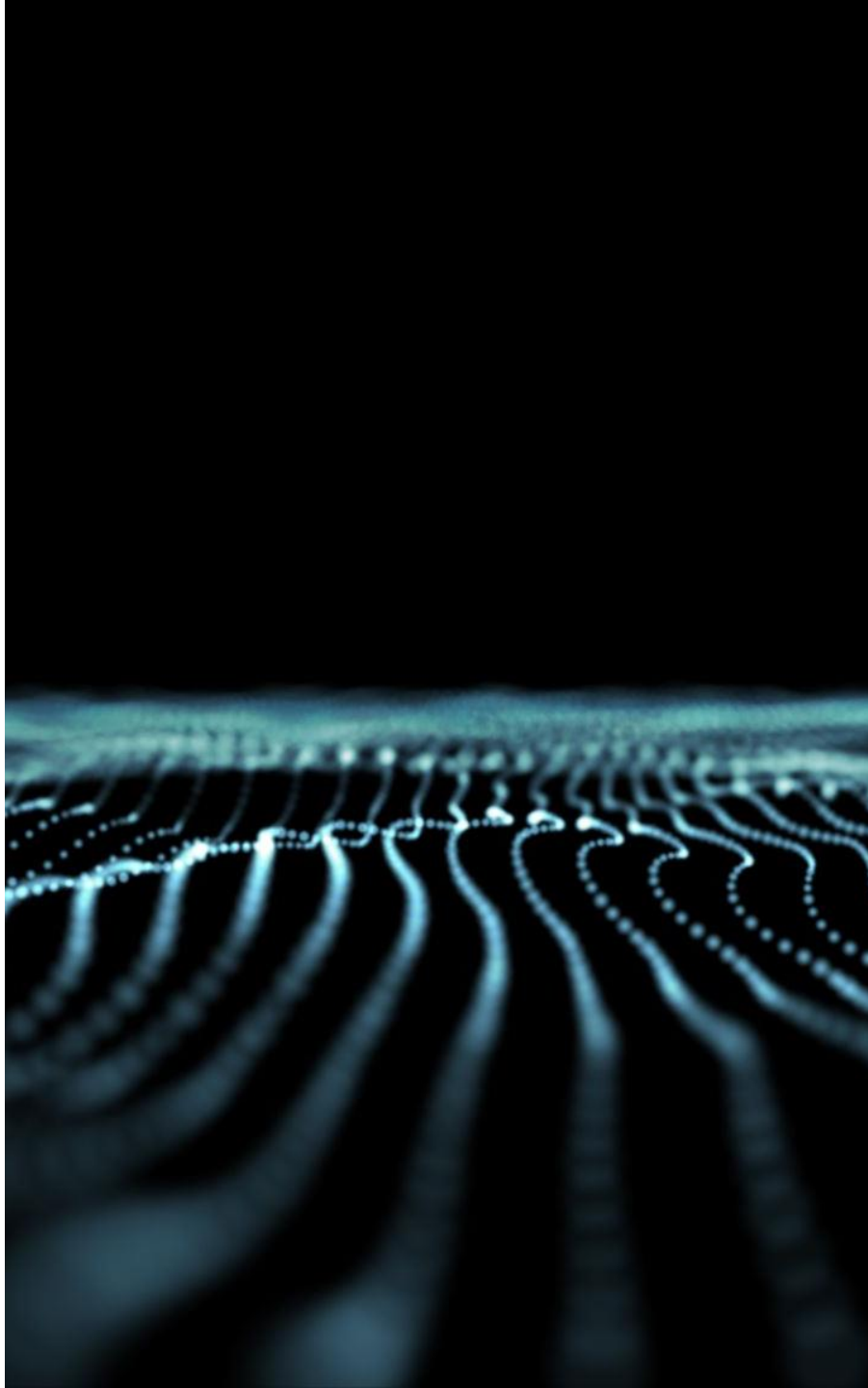
CHATGPT WITH LANGCHAIN (SQL QUERY GENERATOR)



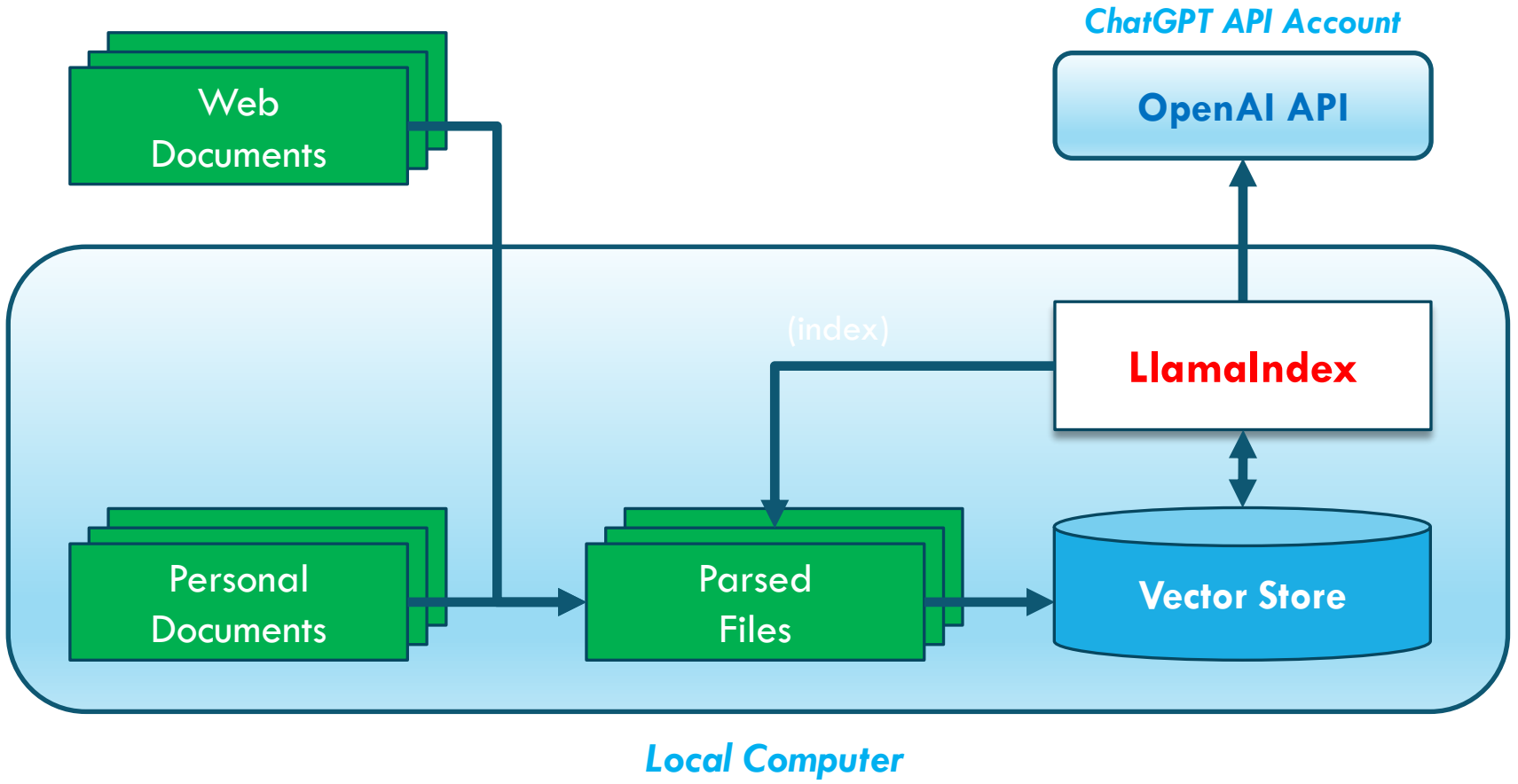

PROJECTS > SKIGEAR_CO_INSPECTION
PREVIEW
Share
Publish
Start training
Training
Documents
Add Question
Add feature name
Ask question...
Provide feature name and question to go further.
SkiGear Co.
1661 Mesa Drive
Las Vegas, Nevada
725-555-5555
MACHINE
SERIAL NUMBER
INSPECTION GRADE
Injection Molder
SGMM-12345
PASS
INSPECTION SUMMARY
The inspection of the injection molder yielded positive results, as no issues or concerns were identified. A meticulous evaluation of the machine's components and operating systems was carried out, confirming their correct functioning. Moreover, a comprehensive examination of the safety measures revealed their full compliance with standards and effective operation. To uphold the machine's optimal condition for future production runs, regular maintenance is advised as a preventive measure. Overall, the inspection affirmed that the machine was operating smoothly, meeting safety regulation, and poised for immediate use.
Injection Unit
GOOD ✓
Mold Clamping Unit
GOOD ✓
Hydraulic System
GOOD ✓
Temperature Control System
GOOD ✓
Ejector System
GOOD ✓
Lubrication System
GOOD ✓
Safety Device
GOOD ✓
Control Software
GOOD ✓
EMILY JOHNSON
Inspected By
2023-04-01
Date

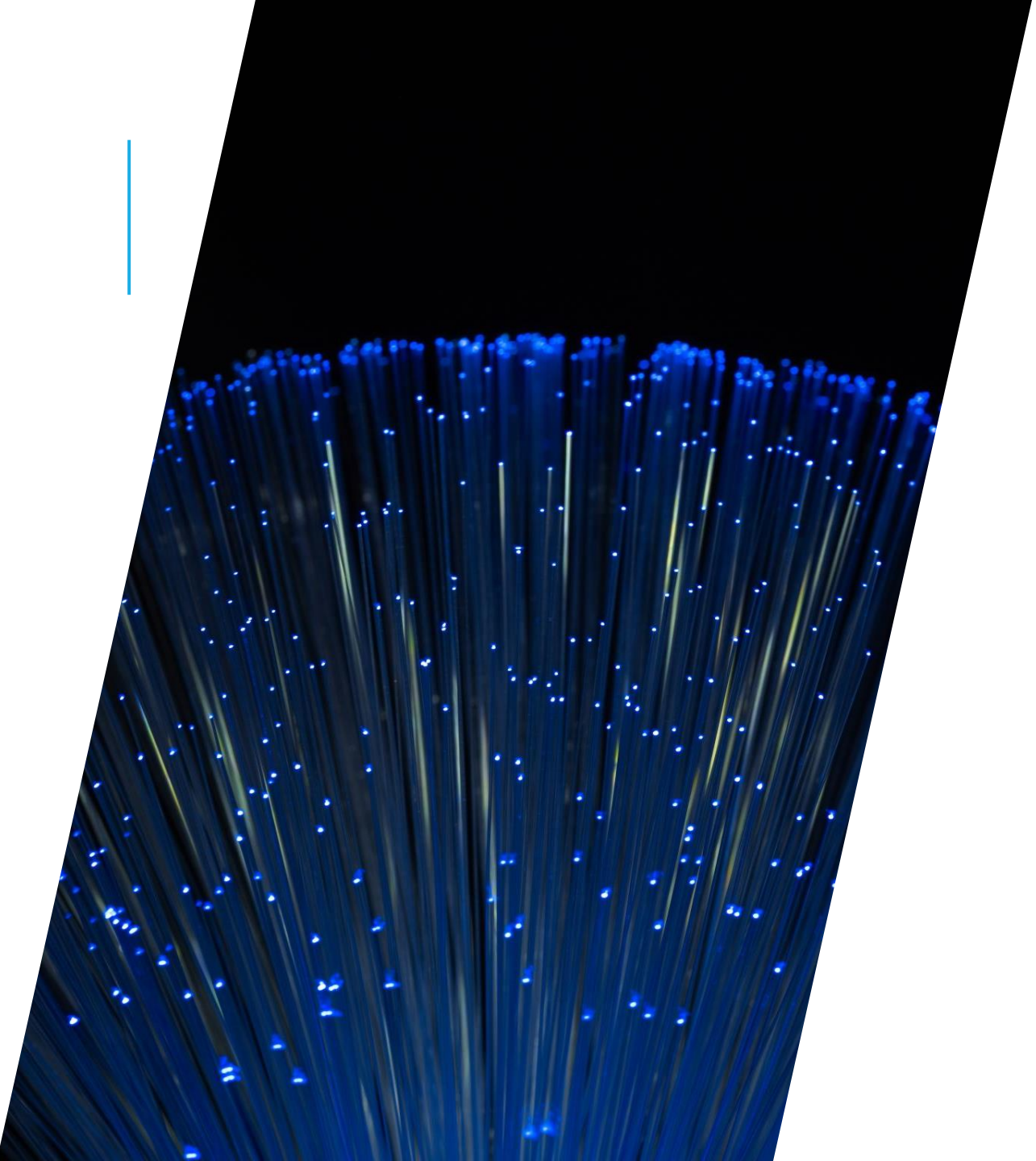
DOCUMENT AI

- **Document AI** = LLM-powered experience for data extraction use cases
- in PuPr soon
- process any doc (pdf, word, txt, screenshots) + get answers to NL questions, w/ a pre-trained model and intuitive interface
- ~LlamaIndex for ChatGPT → create RAG (Retrieval Augmented Generation) KB (~index) for own data
- unstructured data = one of the fastest-growing data types, hard to aggregate/analyze
- Applica = multi-modal LLM for doc intelligence, acquired by Snowflake



CHATGPT WITH LLAMAINDEX (ACCESS PRIVATE UNSTRUCTURED DOCS)





SETUP INSTRUCTIONS

PROJECT SETUP INSTRUCTIONS

SNOWFLAKE ACCOUNT CONFIGURATION

CHATGPT ACCOUNT CONFIGURATION

VSCODE WITH EXTENSIONS