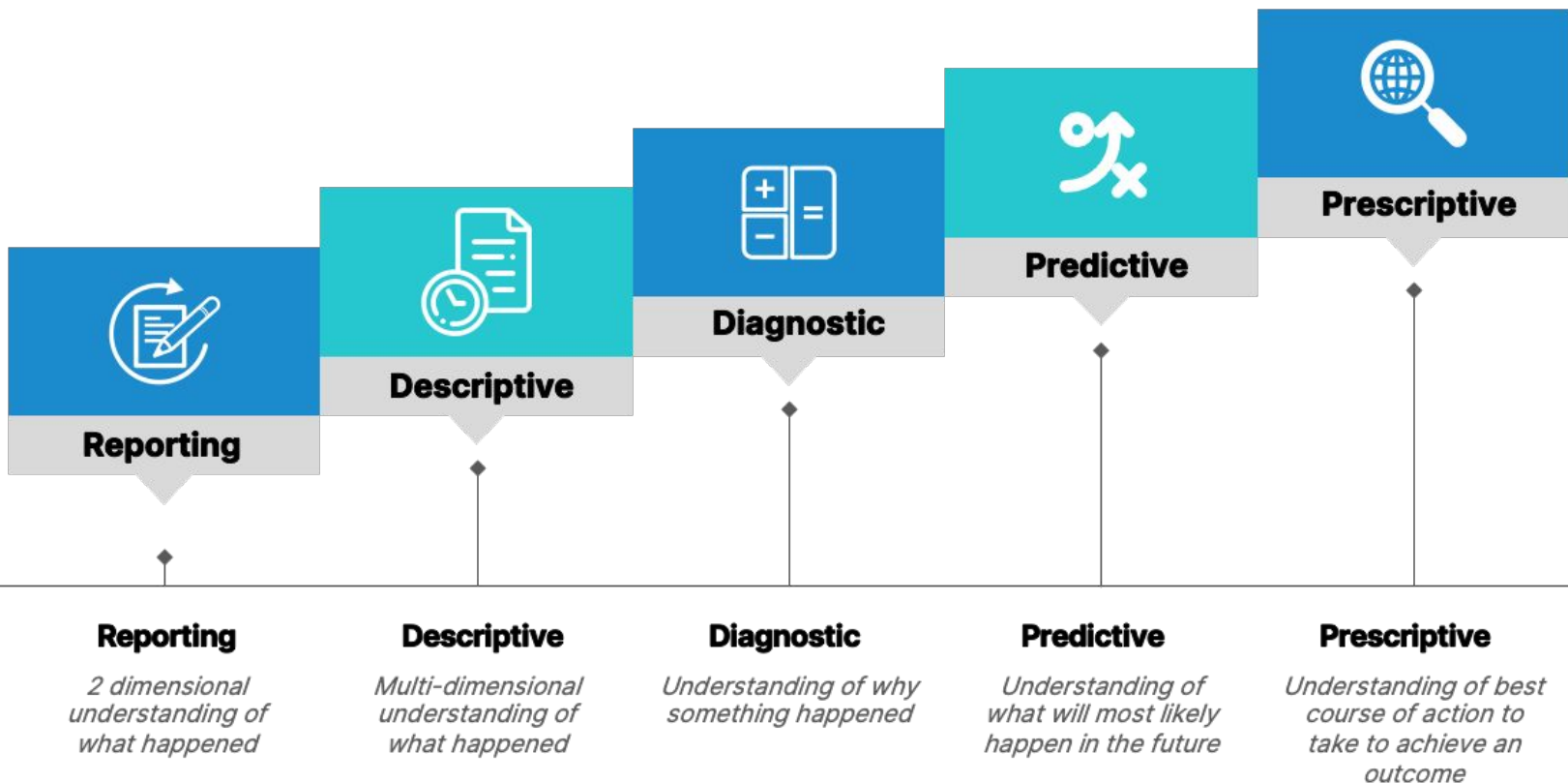# Powering Enterprise Feature Stores With A Universal Semantic Layer

Gaurav Rao, EVP & GM Machine Learning and AI, AtScale

Mature technology companies have a holistic view of data

Reporting	Descriptive	Diagnostic	Predictive	Prescriptive
2 dimensional understanding of what happened	Multi-dimensional understanding of what happened	Understanding of why something happened	Understanding of what will most likely happen in the future	Understanding of best course of action to take to achieve an outcome

Complexity and lack of business adoption prevents ROI and scale-out

$500B in global spend on AI initiatives in 2023 *(IDC)*
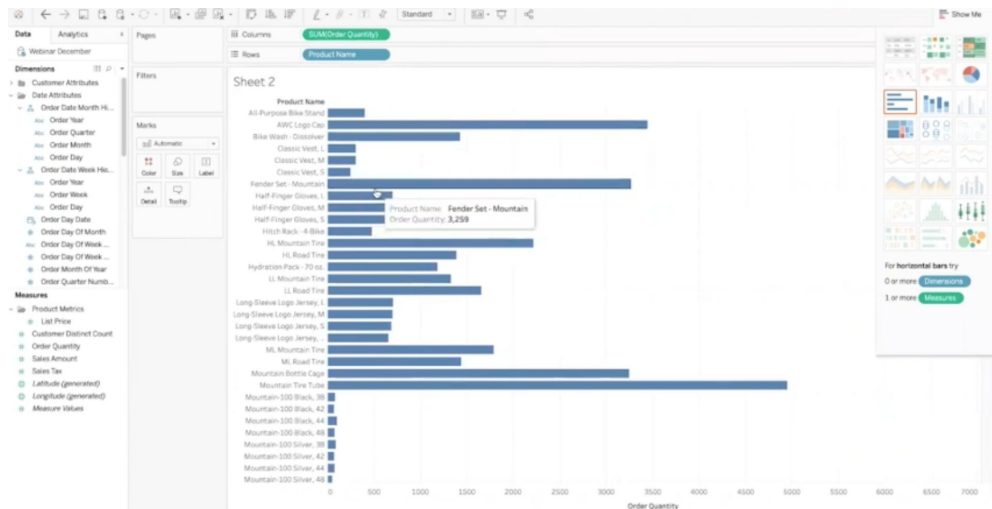
54% of built ML models make it into production *(Gartner)*

90+ days to build a production-ready ML model *(Algorithmia)*

85% of ML models in production fail to produce business value *(Gartner)*

"Features" are not just for data scientists

A universal semantic layer bridges metrics and features
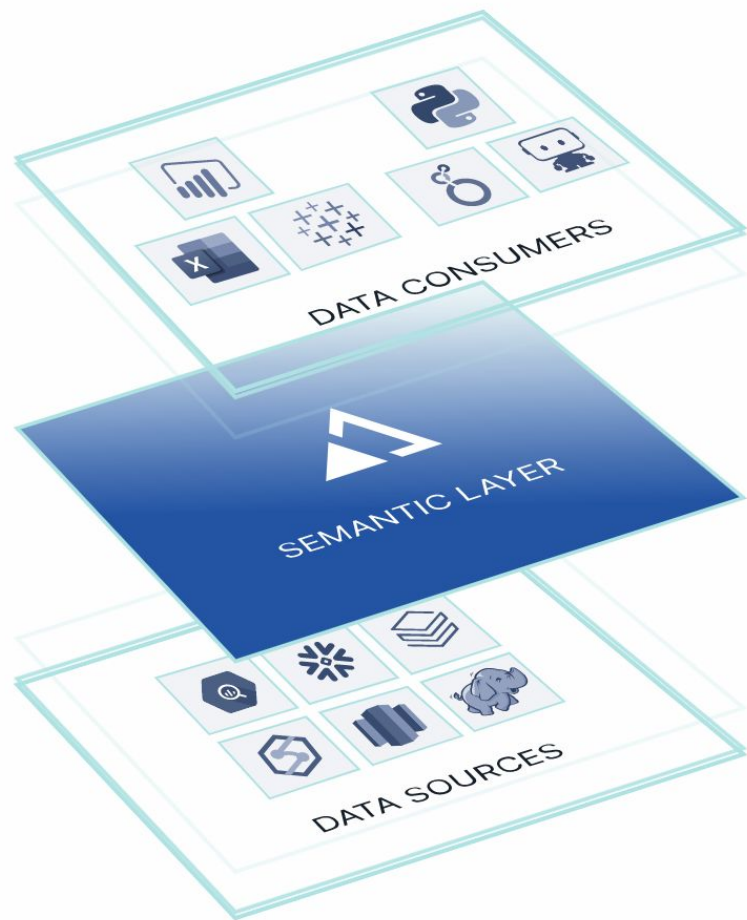
Powering enterprise feature stores with a semantic layer
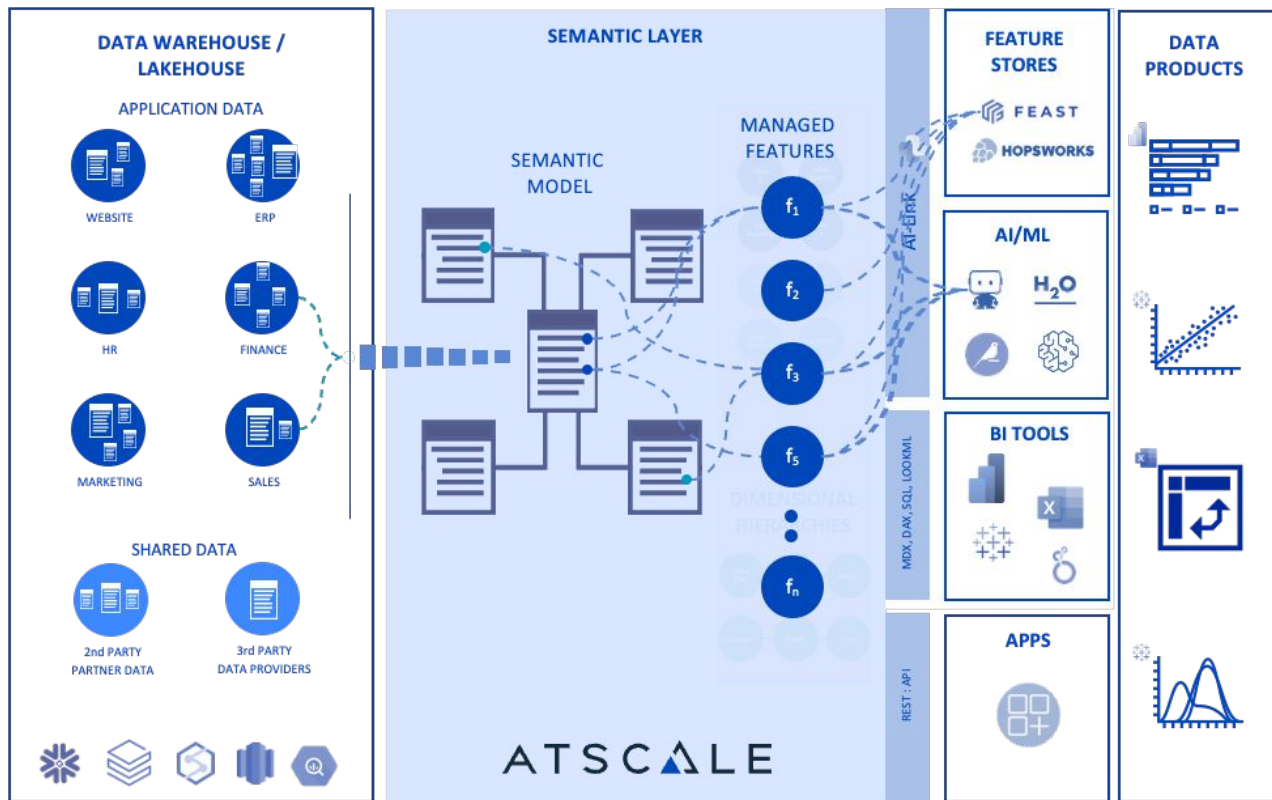
VALUE

- Leverage business-vetted metrics and dimensions to define a set of managed features.

- Expose managed features directly to data scientists and AI platforms via python.

- Aggregate managed features with broader set of features (e.g. streaming) within feature store.

- Serve features directly to AI models or AutoML platforms

*Example:
Forecasting
with a universal
semantic layer
and enterprise
feature store*

Discover business relevant measures from AtScale data model

```
df_ts = data_model.get_data(['date', 'item', 'state', 'category', 'total_units_sold', 'average_units_sold',
                             'sample_standard_deviation_units_sold', 'population_variance_units_sold',
                             'max_units_sold', 'sample_variance_units_sold', 'day_over_day_units_sold',
                             'previous_days_units_sold'])
df_ts.head()
```

Generate time-series features based on those measures

```
df_ts = feature_utils.generate_time_series_features(data_model, df_ts, numeric_features=['total_units_sold'],
                             time_hierarchy='date_hierarchy', level='date',
                             group_features=['state', 'category','item'], shift=7)
df_ts.tail()
```

Train AI model with new time-series features, assess performance

```
from sklearn.model_selection import train_test_split
y_column = 'total_units_sold'
test_df.replace([np.inf, -np.inf], np.nan, inplace=True)
test_df.dropna(how='all')
X, y = (test_df.iloc[:,
                     ~test_df.columns.isin([y_column, 'item', 'category', 'id'])],
        test_df.iloc[:, test_df.columns.isin([y_column])])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
xg_reg = xgb.XGBRegressor()
xg_reg.fit(X_train.iloc[:, ~X_train.columns.isin(['date', 'state'])], y_train)
```

Publish a feature view in a feature store to be used for time series predictions

```
project.add_feature_view(view_name="predictions",
                         project_name='time_series_model',
                         entities=['item', 'state'],
                         timestamp='date',
                         features=['sum_predicted_units_sold',
                                   'total_units_sold',
                                   'total_units_sold_28_day_max'],
                         ttl=datetime.timedelta(weeks=0),)
```

THANK YOU

Learn more:

atscale.com/resources

atscale.com/demo