Assignment: Feature Engineering Using Snowflake and Feature Stores

Doppalapudi Jaya Vardhan

Personal email: [jayavardhand2004@gmail.com](mailto:jayavardhand2004@gmail.com)

Student email: [doppalapudi.jaya2022@vitstudent.ac.in](mailto:doppalapudi.jaya2022@vitstudent.ac.in)

Ph.no: 9154408869

Assignment Info

Feature engineering refers to the transformation of raw data into meaningful and useful representations called features, enhancing the performance of machine learning models.In reality, unprocessed datasets often contain noise, inconsistencies, missing values, or extraneous information that cannot be directly fed into a model. Feature engineering tackles this gap by applying statistical, mathematical, and domain-specific techniques to detect signals that better expose the patterns hidden within the data. In many real-world scenarios, the quality of features outweighs the complexity of the machine learning algorithm—well-designed features can transform a simple model into a highly accurate one.

**Main Things to Take Care**

**Model accuracy**: High-quality features improve the predictive power of ML models.

**Efficiency**: Good features often reduce the need for very complex algorithms.

**Generalization**: Well-structured features allow models to perform better on unseen data.

**Many more as well ,as of my Knowledge I included**

Normalization & Scaling

Numerical attributes in a dataset frequently differ significantly in scale. This may skew machine learning models, as features with greater values might overshadow the learning process. Normalization and scaling modify the numerical ranges of features, ensuring that each feature has an equal impact on the model. Typical methods consist of Min-Max Scaling, which adjusts values to a specified range (commonly 0 to 1), and Z-score Standardization, which centers variables around the average with a standard deviation of one. These methods are especially crucial for algorithms such as K-Nearest Neighbors, Support Vector Machines, and gradient-based optimization techniques.

Encoding Categorical Variables

Machine learning models generally need numerical input, so categorical variables like “gender” or “payment type” need to be transformed into numbers. Encoding converts these variables into formats appropriate for ML models. One-hot encoding transforms each category into a binary column, preventing any implied ordinal relationship, whereas label encoding gives a distinct integer to every category. Correct encoding guarantees that models accurately interpret categorical data without creating deceptive associations among categories.

Identifying and Removing Leaky Features

Data leakage happens when attributes inadvertently reveal details about the target variable that wouldn't be accessible during prediction. Leaky features may cause inflated performance during model training, but they can hinder generalization in actual situations. Recognizing and eliminating these characteristics is essential to avoid deceptive outcomes. Methods involve thorough examination of feature definitions, correlation assessments with the target variable, and validation of domain expertise.

Data Evaluation and Quality Checks

Prior to generating features, it is crucial to evaluate the quality of the original data. This includes managing absent values, identifying outliers, examining feature distributions, and assessing correlations among features. Missing data can be filled in using statistical methods like mean, median, or mode, whereas outliers might need to be removed or transformed. Assessing data quality guarantees that crafted features are dependable and enhance model performance, minimizing the likelihood of noise or unrelated patterns influencing predictions

Time-Based Aggregations and Derived Features

For temporal or behavioral datasets, extracting features from time data can yield important insights. Instances comprise recency, which assesses the days elapsed since a user's last activity, frequency, which tallies occurrences within a specified time frame, and rolling aggregations, like moving averages or cumulative totals. These extracted features reveal trends and patterns across time, allowing models to comprehend sequential or temporal behaviors more effectively.

Intro to Snowfalke

Snowflake is a cloud-driven data warehouse capable of effectively storing structured and semi-structured data. Structured data, including customer transactions or employee records, is kept in conventional tables with specific schemas, whereas semi-structured data such as JSON, Parquet, or XML can be placed in VARIANT columns and accessed directly using SQL. Snowflake facilitates data extraction and preprocessing directly within the warehouse; for instance, SQL queries can summarize transactions to calculate metrics such as total purchases, average amount, and recency for each customer, or transform categorical variables through CASE statements. For machine learning workflows, Snowflake connects effortlessly using Python connectors, Spark, or Databricks, enabling ML models to utilize curated features held in tables or a Feature Store. This method guarantees that data for training and prediction is scalable, reusable, and consistent without transferring large datasets from the warehouse

A screenshot of a computer

AI-generated content may be incorrect.

A Feature Store is a centralized database that maintains curated and reusable features for machine learning algorithms. It guarantees consistency, dependability, and scalability by enabling various models and teams to utilize the same features without the need for recomputation. Feature Stores offer access to both historical (batch) and real-time (online) data, connecting raw information to operational ML workflows. The primary advantages consist of preserving uniform feature logic throughout training and inference, minimizing redundancy in feature engineering tasks, facilitating reusability across various ML projects, and aiding in effective model deployment.

Multiple Feature Store implementations exist, each possessing distinct advantages and disadvantages. AWS SageMaker Feature Store is closely linked with the AWS ecosystem, facilitating online and batch features with automatic versioning, which makes it perfect for organizations that heavily utilize AWS. The Snowflake Feature Store utilizes Snowflake’s unified data warehouse, enabling SQL-driven feature calculations and smooth integration with machine learning processes, although it mainly focuses on batch features and necessitates Snowflake knowledge. Databricks Feature Store closely integrates with Spark and MLflow, providing versioned features that support both batch and real-time processing, making it ideal for teams utilizing Databricks for ETL and ML processes. Selecting the appropriate Feature Store is influenced by elements like cloud environment, real-time feature needs, and the organization's current data framework.

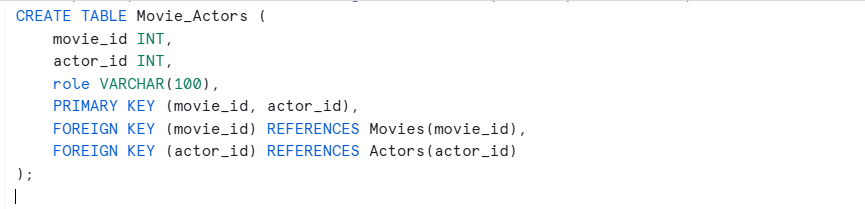
|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Store** | **Key Strengths** | **Limitations** | **Use Case** |
| **AWS SageMaker Feature Store** | Native AWS integration, supports online & batch features, automatic versioning | Limited to AWS ecosystem | Organizations fully using AWS for ML |
| **Snowflake Feature Store** | Centralized storage for structured & semi-structured data, SQL-based feature computation, seamless ML integration | Primarily batch features, requires Snowflake expertise | Enterprises using Snowflake for data warehousing |
| **Databricks Feature Store** | Tight Spark/MLflow integration, supports real-time & batch features, versioned features | Requires Databricks environment, higher cost at scale | Teams using Databricks for ETL and ML workflows |

SQL DATABASE

DDL

A white background with blue text

AI-generated content may be incorrect.



Results:

A screenshot of a computer

AI-generated content may be incorrect.

Insertion

A screenshot of a computer

AI-generated content may be incorrect.

Results:

A white paper with black lines

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

After feature engineering, the computed features can be **stored in a Feature Store**, which acts as a centralized and versioned repository for all machine learning features. In the case of MovieDB, aggregated features such as the number of actors in a movie, average actor age, and number of lead roles can be stored in a dedicated table like FEATURE\_STORE.MOVIE\_FEATURES. Each feature is saved along with metadata such as its data type, computation logic, and update frequency, which ensures that all ML models access consistent and reliable data. Storing features in a Feature Store reduces duplication of feature engineering, improves reproducibility, and allows multiple models to reuse the same high-quality features efficiently.

The ML workflow for MovieDB begins with **data extraction**, where raw movie, actor, and transaction data are pulled from the database. SQL queries are used to join tables like Movies, Actors, and Movie\_Actors to retrieve relevant data for modeling.

Next is **feature engineering**, which transforms raw data into machine learning-ready features. Examples include aggregating the number of actors per movie, calculating the average age of actors, and counting the number of lead roles. Categorical variables like genre can be encoded using one-hot or label encoding, and numerical features can be normalized or scaled. This ensures the features are consistent, meaningful, and free from leaks.

The engineered features are then **stored in a Feature Store**, such as a dedicated Snowflake table FEATURE\_STORE.MOVIE\_FEATURES. Storing features centrally provides versioning, reusability, and ensures that all ML models access the same high-quality data. Metadata, such as computation logic and update frequency, is maintained alongside the features to improve reproducibility and traceability.

For **model training**, features are retrieved from the Feature Store using SQL or Python connectors. For example, a Pandas dataframe can be loaded with the features, which are then split into training and test sets. Machine learning algorithms, such as regression to predict movie ratings or classification to predict genre, can be applied.

Feature Engineering Process

For predicting sales price, I first looked at the STORE\_SALES table in Snowflake and picked features that I thought would influence the target. I chose item\_sk because different products have different prices, store\_sk because prices can vary between stores, and quantity since more units sold usually means a higher sales price. I noticed that some values were missing, so I removed rows with nulls in these key columns to avoid errors in my models. I also created new features to give the model more insight: for example, I calculated the average quantity sold per item and per store to capture trends, and I could extract month or day-of-week from the sale date to account for seasonal effects. I made sure not to include any leaky features that would give away the target. After this feature engineering, I had a clean, meaningful set of inputs that I could feed into my models to help them learn the patterns in sales price.

After training multiple models on my engineered features, I learned a lot about how different factors affect sales price. The ensemble models, like Random Forest and Gradient Boosting, performed really well and captured non-linear relationships between the features and the target. Linear Regression worked okay, but it couldn’t capture the more complex interactions as effectively. I saw that quantity had the strongest impact on sales price, which makes sense because selling more units naturally increases revenue. The encoded item\_sk and store\_sk features also contributed, showing that some products and stores consistently influence sales differently. Visualizing the predictions versus actual values confirmed that most predictions were very close to the real sales prices, and the error distributions were centered around zero, which means the models are reliable. Overall, I gained a good understanding of which features are important and how different models can learn patterns in retail sales data.

A screenshot of a computer screen

AI-generated content may be incorrect.

A graph of sales

AI-generated content may be incorrect.

A graph of error distribution

AI-generated content may be incorrect.

A graph of sales and sales

AI-generated content may be incorrect.

A graph of error distribution

AI-generated content may be incorrect.