codeDataEnrichmentandFeatureEngineering

October 8, 2024

10.1 Data Enrichment and Feature Engineering

Data enrichment and feature engineering are crucial steps in the data analysis pipeline, aimed at enhancing the quality and predictive power of a dataset. These processes involve adding additional information to the dataset, creating new variables (features), and transforming existing ones to better capture the underlying patterns in the data. Data enrichment brings in external or supplementary data, while feature engineering transforms this and existing data into a format that can improve the performance of predictive models and analysis.

In today's data-driven world, the raw data collected from various sources often lacks the depth or context needed for comprehensive analysis. By enriching data and engineering features, analysts can uncover hidden relationships, enhance model accuracy, and derive more meaningful insights. This chapter delves into the strategies and techniques for effectively enriching data and engineering features to maximize their value in the analytical process.

Definition

Data Enrichment: The process of enhancing existing data by incorporating additional information from external or supplementary sources. This can involve appending new columns with related data, such as demographics, geographic information, or transactional history, to provide more context and depth.

Feature Engineering: The practice of creating new features (variables) or transforming existing ones to improve the predictive power of models. This can include deriving new features from raw data, encoding categorical variables, creating interaction terms, or applying mathematical transformations.

Objective

The primary objectives of data enrichment and feature engineering are:

1. Enhancing Data Quality: By enriching data, we add valuable context that enhances the quality and completeness of the dataset.
2. Improving Predictive Models: Feature engineering helps in creating more informative features that can improve the performance of machine learning models.
3. Uncovering Hidden Patterns: These processes help in identifying and leveraging hidden patterns within the data that might not be apparent from the raw data.
4. Increasing Model Interpretability: Carefully engineered features can make models more interpretable, as they often align more closely with the problem domain.

Importance

Data enrichment and feature engineering are vital for several reasons:

1. Boosting Model Accuracy: Well-engineered features can significantly improve the accuracy and robustness of predictive models.
2. Providing Contextual Insights: Enriched data offers additional context, leading to more nuanced and actionable insights.
3. Addressing Data Gaps: Enrichment helps in filling in the gaps within the existing dataset, leading to a more comprehensive analysis.
4. Enabling Complex Analysis: Feature engineering allows for the transformation of raw data into formats that can be used in complex analyses, such as machine learning or statistical modeling.

10.2 Techniques List and Definitions 1. Creating Interaction Features: Combining two or more features to capture the interaction between them. 2. Adding External Data: Enriching the dataset with external sources, such as demographic information, weather data, or economic indicators. 3. Aggregating Features: Summarizing data through aggregations like mean, sum, or count based on different categories or time periods. 4. Creating Lag Features: Creating features based on previous time steps in time series data. 5. Calculating Rolling Statistics: Using moving averages, sums, or other statistics calculated over a rolling window. 6. Binning Continuous Variables: Converting continuous variables into categorical bins to capture non-linear relationships. 7. Polynomial Feature Creation: Generating polynomial features to capture non-linear relationships between variables. 8. Text Feature Extraction: Extracting meaningful features from text data, such as word counts or sentiment scores. 9. Encoding Cyclical Features: Encoding cyclical features, such as time or geographic data, to capture periodicity. 10. Dimensionality Reduction Techniques: Using techniques like PCA to reduce the number of features while retaining essential information.

10.2.1 Creating Interaction Features

Interaction features capture the combined effect of two or more features, which may not be apparent when considering the features individually.

[11]:
```
import pandas as pd

# Sample Data
data = {
    'Feature1': [10, 20, 30, 40, 50],
    'Feature2': [1, 2, 3, 4, 5]
}
df = pd.DataFrame(data)

# Creating Interaction Feature
df['Feature1_Feature2_Interaction'] = df['Feature1'] * df['Feature2']

print(df)
```

```
   Feature1  Feature2  Feature1_Feature2_Interaction
0        10         1                             10
1        20         2                             40
2        30         3                             90
3        40         4                            160
4        50         5                            250
```

Explanation

Interaction Feature: The new feature, Feature1_Feature2_Interaction, is created by multiplying Feature1 and Feature2. Purpose: This feature may capture interactions that are important for predicting the target variable, improving model accuracy.

10.2.2 Adding External Data

Adding external data can enrich the existing dataset with additional context, leading to better predictive performance.

[12]:
```python
import pandas as pd

# Sample Data
data = {
    'Product ID': [1, 2, 3, 4, 5],
    'Sales': [100, 150, 200, 130, 170]
}
df = pd.DataFrame(data)

# External Data - Adding weather information
external_data = {
    'Product ID': [1, 2, 3, 4, 5],
    'Average_Temperature': [30, 25, 27, 32, 28]
}
external_df = pd.DataFrame(external_data)

# Merging External Data
df = df.merge(external_df, on='Product ID', how='left')

print(df)
```

```
   Product ID  Sales  Average_Temperature
0           1    100                   30
1           2    150                   25
2           3    200                   27
3           4    130                   32
4           5    170                   28
```

Explanation

External Data: The dataset is enriched with weather information, which might be relevant for predicting sales. Merging: The merge function is used to combine the existing dataset with the external data based on a common key (Product ID).

10.2.3 Aggregating Features

Aggregating features involves summarizing data through various statistical measures, such as mean, sum, or count, often based on groupings or time periods.

[13]:
```python
import pandas as pd

# Sample Data
data = {
    'Customer ID': [1, 1, 2, 2, 3],
    'Purchase Amount': [100, 200, 150, 250, 300]
}
df = pd.DataFrame(data)

# Aggregating Purchase Amount by Customer ID
df_aggregated = df.groupby('Customer ID')['Purchase Amount'].sum().reset_index()

print(df_aggregated)
```

```
   Customer ID  Purchase Amount
0            1              300
1            2              400
2            3              300
```

Explanation

Aggregation: The groupby function is used to sum the Purchase Amount for each Customer ID, providing insight into total spending by each customer. Purpose: Aggregated features can simplify the dataset and reveal patterns or trends.

10.2.4 Creating Lag Features

Lag features involve using past values of a variable as predictors in time series forecasting.

[14]:
```python
import pandas as pd

# Sample Data
data = {
    'Date': pd.date_range(start='2023-01-01', periods=5, freq='D'),
    'Sales': [100, 150, 130, 170, 160]
}
df = pd.DataFrame(data)

# Creating Lag Feature
df['Sales_Lag1'] = df['Sales'].shift(1)

print(df)
```

```
        Date  Sales  Sales_Lag1
0 2023-01-01    100         NaN
1 2023-01-02    150       100.0
2 2023-01-03    130       150.0
3 2023-01-04    170       130.0
4 2023-01-05    160       170.0
```

Explanation

Lag Feature: The new feature Sales_Lag1 contains the sales values from the previous day. Purpose: Lag features are essential in time series analysis to capture temporal dependencies.

10.2.5 Calculating Rolling Statistics

Rolling statistics are computed over a specified window of past observations, smoothing out short-term fluctuations.

[15]:
```
import pandas as pd

# Sample Data
data = {
    'Date': pd.date_range(start='2023-01-01', periods=5, freq='D'),
    'Sales': [100, 150, 130, 170, 160]
}
df = pd.DataFrame(data)

# Calculating Rolling Mean
df['Rolling_Mean'] = df['Sales'].rolling(window=3).mean()

print(df)
```

```
        Date  Sales  Rolling_Mean
0 2023-01-01    100           NaN
1 2023-01-02    150           NaN
2 2023-01-03    130    126.666667
3 2023-01-04    170    150.000000
4 2023-01-05    160    153.333333
```

Explanation

Rolling Mean: This feature calculates the average sales over a 3-day window. Purpose: Rolling statistics help smooth out noise in time series data and reveal underlying trends.

10.2.6 Binning Continuous Variables

Binning involves converting continuous variables into discrete categories or bins, which can help in capturing non-linear relationships.

[16]:
```
import pandas as pd

# Sample Data
data = {'Age': [22, 35, 58, 45, 29]}
df = pd.DataFrame(data)

# Binning Age into categories
bins = [0, 25, 45, 60]
labels = ['Youth', 'Adult', 'Senior']
df['Age Group'] = pd.cut(df['Age'], bins=bins, labels=labels)

print(df)
```

```
   Age Age Group
0   22     Youth
1   35     Adult
2   58    Senior
3   45     Adult
4   29     Adult
```

Explanation

Binning: The pd.cut function is used to categorize ages into three groups: ‘Youth’, ‘Adult’, and ‘Senior’. Purpose: Binning helps in reducing the impact of outliers and capturing non-linear effects.

10.2.7. Polynomial Feature Creation

Polynomial features are created by raising existing features to a power, which allows the model to capture non-linear relationships.

[17]:
```
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures

# Sample Data
data = {'Feature1': [1, 2, 3, 4, 5]}
df = pd.DataFrame(data)

# Creating Polynomial Features
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(df[['Feature1']])

df_poly = pd.DataFrame(poly_features, columns=['Feature1', 'Feature1_squared'])
print(df_poly)
```

```
   Feature1  Feature1_squared
0       1.0               1.0
1       2.0               4.0
2       3.0               9.0
3       4.0              16.0
4       5.0              25.0
```

Explanation

Polynomial Features: The original feature Feature1 is squared to create a new feature Feature1_squared. Purpose: Polynomial features enable the model to fit more complex, non-linear patterns.

10.2.8. Text Feature Extraction

Text feature extraction involves converting unstructured text data into meaningful numerical features for analysis.

[18]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
```

```
# Sample Data
data = {'Text': ['I love data science', 'Data science is great', 'I love␣
  ↪learning new things']}
df = pd.DataFrame(data)

# Text Feature Extraction using CountVectorizer
vectorizer = CountVectorizer()
text_features = vectorizer.fit_transform(df['Text']).toarray()

df_text_features = pd.DataFrame(text_features, columns=vectorizer.
  ↪get_feature_names_out())
print(df_text_features)
```

```
   data  great  is  learning  love  new  science  things
0     1      0   0         0     1    0        1       0
1     1      1   1         0     0    0        1       0
2     0      0   0         1     1    1        0       1
```

Explanation

CountVectorizer: This technique converts text data into a matrix of token counts, creating features based on word frequency. Purpose: Extracted features from text data are used in various machine learning models to analyze textual information.

10.2.9. Encoding Cyclical Features

Cyclical features represent variables with periodic patterns, such as time or geographic data, and are encoded to capture their cyclic nature.

[19]:
```
import pandas as pd
import numpy as np

# Sample Data
data = {'Hour': [0, 6, 12, 18, 23]}
df = pd.DataFrame(data)

# Encoding Hour as Cyclical Feature
df['Hour_sin'] = np.sin(2 * np.pi * df['Hour'] / 24)
df['Hour_cos'] = np.cos(2 * np.pi * df['Hour'] / 24)

print(df)
```

```
   Hour      Hour_sin      Hour_cos
0     0  0.000000e+00  1.000000e+00
1     6  1.000000e+00  6.123234e-17
2    12  1.224647e-16 -1.000000e+00
3    18 -1.000000e+00 -1.836970e-16
4    23 -2.588190e-01  9.659258e-01
```

Explanation

Cyclical Encoding: The Hour feature is transformed into two new features using sine and cosine functions to represent its cyclical nature. Purpose: Encoding cyclical features helps capture patterns that repeat over time, improving model predictions.

10.2.10 Dimensionality Reduction Techniques

Dimensionality reduction techniques reduce the number of features while preserving essential information, which helps in avoiding overfitting and improving model performance.

[20]:
```
import pandas as pd
from sklearn.decomposition import PCA

# Sample Data
data = {
    'Feature1': [1, 2, 3, 4, 5],
    'Feature2': [2, 4, 6, 8, 10],
    'Feature3': [5, 7, 9, 11, 13]
}
df = pd.DataFrame(data)

# Applying PCA for Dimensionality Reduction
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(df)

df_reduced = pd.DataFrame(reduced_features, columns=['PC1', 'PC2'])
print(df_reduced)
```

```
   PC1           PC2
0  6.0  4.282044e-16
1  3.0 -1.427348e-16
2 -0.0 -0.000000e+00
3 -3.0  1.427348e-16
4 -6.0  2.854696e-16
```

Explanation

PCA: Principal Component Analysis (PCA) is used to reduce the dataset to two principal components (PC1 and PC2), which explain the maximum variance in the data. Purpose: Dimensionality reduction simplifies the dataset and helps improve model generalization by focusing on the most important features.