Feature Engineering Ultimate Cheat Sheet

Transforming Raw Data into Powerful Predictors

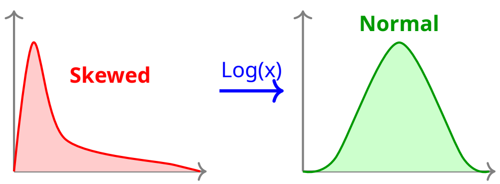
1 Engineering Workflow

Process Overview

Goal: Create features that correlate strongly with the target variable, remove noise, and ensure data meets model assumptions (e.g., normality).

2 1. Variable Transformation

Log Transformation *Use for right-skewed data (e.g., Income, Price). Compresses large values.*




```
import numpy as np
# Add 1 to handle 0 values
df['log_price'] = np.log1p(df['price'])
```

Box-Cox / Yeo-Johnson *Generalizes power transformations to fix skewness and variance.*

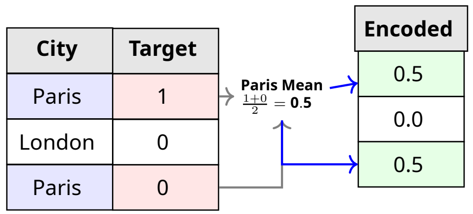
```
from scipy.stats import boxcox
# Data must be positive
df['boxcox'], _ = boxcox(df['pos_col'])
```

Binning (Discretization) *Converts continuous numerical data into categorical buckets. handles outliers.*

```
# Fixed Width
df['age_bin'] = pd.cut(df['age'], bins=4)

# Quantile Based (Equal Frequency)
df['price_q'] = pd.qcut(df['price'], q=4)
```

3 2. Categorical Encoding

Target Encoding (Mean Encoding) *Replaces category with the average target value. Powerful but risky (overfitting).*




```
# Calculate mean target per category
means = df.groupby('city')['target'].mean()
df['city_enc'] = df['city'].map(means)
```

Frequency Encoding *Replaces category with its count/frequency.*

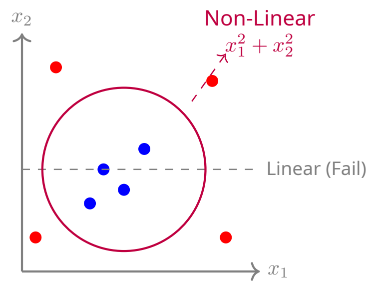
```
freq = df['color'].value_counts()
df['color_freq'] = df['color'].map(freq)
```

One-Hot vs Label Encoding

- **One-Hot**: For Nominal data (Red, Blue). `pd.get_dummies()`.
- **Label**: For Ordinal data (Low, Med, High). `LabelEncoder`.

4 3. Feature Creation

Polynomial Features *Creates non-linear relationships (x^2, xy, y^2).*




```
from sklearn.preprocessing import \
    PolynomialFeatures

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
```

Domain Specific Interactions

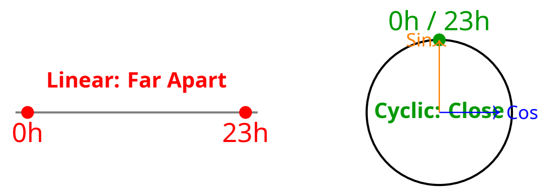
```
# Ratio features
df['price_per_sqft'] = df['price'] / df['sqft']

# Sum/Diff
df['total_income'] = df['applicant'] + \
                     df['spouse']
```

5 4. DateTime & Cyclic

Basic Extraction

```
df['dt'] = pd.to_datetime(df['date'])
df['year'] = df['dt'].dt.year
df['dow'] = df['dt'].dt.dayofweek
```

Cyclic Encoding (Sin/Cos) *Preserves the "closeness" of Hour 23 and Hour 0.*




```
import numpy as np

# Normalize hour 0-23 to 0-2pi
df['hour_sin'] = np.sin(2*np.pi*df['hour']/24)
df['hour_cos'] = np.cos(2*np.pi*df['hour']/24)
```

6 5. Text Features (NLP)

Simple Stats

```
df['word_count'] = df['text'].apply(
    lambda x: len(str(x).split()))
df['char_len'] = df['text'].str.len()
```

TF-IDF (Term Frequency - IDF) *Reflects how important a word is to a document in a collection.*

```
from sklearn.feature_extraction.text \
    import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=100)
X_text = tfidf.fit_transform(df['text'])
```

7 6. Feature Selection

Filter Methods *Statistical tests (Correlation, Chi-Square).*

```
# Remove high correlation features
corr = df.corr().abs()
upper = corr.where(np.triu(np.ones(corr.shape), k=1).
    astype(bool))
to_drop = [c for c in upper.columns if any(upper[c] >
    0.95)]
```

Wrapper Methods (RFE) *Recursive Feature Elimination. Iteratively trains model and removes weakest features.*

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

selector = RFE(LinearRegression(), n_features_to_select=5)
selector = selector.fit(X, y)
selected_cols = X.columns[selector.support_]
```

Embedded Methods (Lasso/Tree) *Model selects features during training.*

```
from sklearn.ensemble import \
    RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(X, y)
importances = pd.Series(
    rf.feature_importances_, index=X.columns
).sort_values(ascending=False)
```