Feature Engineering Report

Prepared by: Shahzain Mirza

## **Table of Contents**

1. Introduction

2. Tools and Libraries Used

3. Module 1: Handling Missing Values

4. Module 2: Encoding Categorical Variables

5. Module 3: Feature Scaling

6. Module 4: Polynomial and Interaction Features

7. Module 5: Binning and Feature Extraction

8. Module 6: Log and Power Transformation

9. Module 7: Target Encoding and Geospatial Features

10. Module 8: Dimensionality Reduction (PCA)

11. Conclusion

## **1. Introduction**

Feature engineering is a pivotal step in the machine learning pipeline, directly influencing model performance. This report explores fundamental and advanced techniques for transforming raw data into meaningful features, leveraging Python and its powerful libraries.

**2. Tools and Libraries Used**

* Python
* NumPy: For numerical operations.
* Pandas: For data manipulation.
* Scikit-learn: For machine learning and preprocessing.

**3. Module 1: Handling Missing Values**

Missing data can distort analysis and model performance. Common techniques include:

* Deletion: Remove rows/columns with missing values.
* Imputation: Fill gaps using mean, median, or mode.

Example:

python

df['age'] = df['age'].fillna(df['age'].mean()) # Mean imputation

df = df.dropna() # Deletion

## **4. Module 2: Encoding Categorical Variables**

Categorical data must be converted to numerical formats for modeling:

One-Hot Encoding: Creates binary columns for each category.

Label Encoding: Assigns integer labels to categories.

Example:

python

df = pd.get\_dummies(df, columns=['gender']) # One-Hot Encoding

le = LabelEncoder()

df['gender\_encoded'] = le.fit\_transform(df['gender']) # Label Encoding

## **5. Module 3: Feature Scaling**

Features on different scales can bias models. Scaling standardizes their range:

StandardScaler: Normalizes to zero mean and unit variance.

Example

python

scaler = StandardScaler()

df\_scaled = scaler.fit\_transform(df)

## **6. Module 4: Polynomial and Interaction Features**

New features can capture nonlinear relationships:

PolynomialFeatures: Generates polynomial combinations.

Example:

python

poly = PolynomialFeatures(degree=2)

df\_poly = poly.fit\_transform(df)

## **7. Module 5: Binning and Feature Extraction**

Binning: Groups continuous data into intervals.

Feature Extraction: Derives features from text/dates.

Example:

python

df['age\_bin'] = pd.cut(df['age'], bins=[0, 30, 60, 90]) # Binning

df['text\_len'] = df['text'].apply(len) # Text feature extraction

**8. Module 6: Log and Power Transformation**

Reduces skewness in data distributions:

Log Transformation: Compresses large values.

Example:

python

df['log\_income'] = np.log(df['income'])

pt = PowerTransformer()

df\_transformed = pt.fit\_transform(df[['income']])

## **9. Module 7: Target Encoding and Geospatial Features**

Target Encoding: Replaces categories with target statistics.

Geospatial Features: Calculates distances between coordinates.

Example:

python

df['city\_encoded'] = df['city'].map(df.groupby('city')['target'].mean())

from geopy.distance import geodesic

distance = geodesic(point1, point2).km

## **10. Module 8: Dimensionality Reduction (PCA)**

Reduces feature count while preserving variance:

Example:

python

pca = PCA(n\_components=2)

df\_pca = pca.fit\_transform(scaler.fit\_transform(df))

## **11. Conclusion**

Feature engineering is the backbone of effective machine learning. By applying these techniques—handling missing values, encoding, scaling, and more - we enhance model accuracy and robustness. Mastery of these methods ensures data is optimally prepared for modeling.