Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Fall, Year:2025), B.Sc. in CSE (Day)

Lab Report No: 1
Course Title: Data Mining Lab
Course Code: CSE-436 Section:222-D1

Lab Experiment Name: Feature Engineering techniques in Data Mining

Student Details

	Name	ID
1.	Md. Abdullah Al Moin	222902070

Submission Date : 13-10-2025
Course Teacher's Name : Md. Atikuzzaman

Lab Report Status
Marks: ………………………………… Signature:.....................
Comments:... Date:..............................

TITLE OF THE LAB REPORT EXPERIMENT:

Feature Engineering Techniques In Data Mining.

OBJECTIVES:

The main objective of this lab is to apply data preprocessing techniques such as imputation, outlier detection, feature engineering, feature scaling, and encoding. It also aims to build an automated workflow using pipelines to improve model performance and ensure efficient, consistent data handling during machine learning tasks.

PROCEDURE:

1. Load the dataset and inspect missing values.
2. Handle missing data using median imputation.
3. Encode categorical features with OneHotEncoder.
4. Split the dataset into training and testing sets.
5. Apply feature scaling for normalization.
6. Detect and remove outliers if needed.
7. Create a pipeline combining preprocessing and model training.

IMPLEMENTATION&TEST RESULT:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/DataMining/oceanvoyageData.csv")
```

```
!pip install pandas numpy scikit-learn matplotlib seaborn -q

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, PolynomialFeatures, KBinsDiscretizer, FunctionTransformer
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings('ignore')

print(f"Pandas Version: {pd.__version__}")
print(f"NumPy Version: {np.__version__}")
print(f"Scikit-learn Version: {sklearn.__version__}")
```

```
Pandas Version: 2.2.2
NumPy Version: 2.0.2
Scikit-learn Version: 1.6.1
```

```
df.head()
```

	Age	Fare	TravelClass	Gender	EmbarkedPort	SibSp	ParCh	Survived
0	28	231.376717	2	F	PortC	3	1	0
1	40	18.095096	3	M	PortB	1	0	1
2	9	111.537253	3	M	PortC	0	1	0
3	20	104.713824	3	F	PortC	1	0	0
4	28	NaN	2	F	PortA	1	1	0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          1000 non-null   int64
 1   Fare         960 non-null    float64
 2   TravelClass  1000 non-null   int64
 3   Gender       1000 non-null   object
 4   EmbarkedPort 1000 non-null   object
 5   SibSp        1000 non-null   int64
 6   ParCh        1000 non-null   int64
 7   Survived     1000 non-null   int64
dtypes: float64(1), int64(5), object(2)
memory usage: 62.6+ KB
```

```
df.describe()
```

	Age	Fare	TravelClass	SibSp	ParCh	Survived
count	1.000000e+03	960.000000	1000.000000	1000.00000	1000.000000	1000.000000
mean	-4.611686e+17	93.505351	2.323000	1.21600	0.792000	0.349000
std	2.011193e+18	163.646527	0.788221	1.13958	0.918466	0.476892
min	-9.223372e+18	0.162020	1.000000	0.00000	0.000000	0.000000
25%	1.900000e+01	23.358816	2.000000	0.00000	0.000000	0.000000
50%	2.800000e+01	56.915283	3.000000	1.00000	1.000000	0.000000
75%	3.800000e+01	111.418688	3.000000	2.00000	1.000000	1.000000
max	1.100000e+02	2645.627902	3.000000	6.00000	5.000000	1.000000

```
X = df.drop(['Survived'], axis=1)
y = df['Survived']

df.columns = df.columns.str.lower()
X.columns = X.columns.str.lower()
y.name = y.name.lower()

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")

numeric_features_base = ['age', 'fare', 'sibsp', 'parch', 'travelclass']
categorical_features_base = ['gender', 'embarkedport']

preprocessor_base = ColumnTransformer(transformers=[
    ('num', SimpleImputer(strategy='median'), numeric_features_base),
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features_base)
])

baseline_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor_base),
    ('classifier', LogisticRegression(random_state=42))
])

baseline_pipeline.fit(X_train, y_train)
y_pred_base = baseline_pipeline.predict(X_test)
baseline_accuracy = accuracy_score(y_test, y_pred_base)

print(f"\nBaseline Model Accuracy: {baseline_accuracy:.4f}")
```

```
Training data shape: (800, 7)
Testing data shape: (200, 7)

Baseline Model Accuracy: 0.3850
```

```python
print("\n--- Part 2: Creating Features from Domain Knowledge ---")

X_train_eng = X_train.copy(deep=True)

X_train_eng['FamilySize'] = X_train_eng['sibsp'] + X_train_eng['parch'] + 1

X_train_eng['IsAlone'] = 0
X_train_eng.loc[X_train_eng['FamilySize'] == 1, 'IsAlone'] = 1

X_train_eng['FarePerPerson'] = X_train_eng['fare'] / X_train_eng['FamilySize']

X_train_eng['AgeGroup'] = pd.cut(
    X_train_eng['age'],
    bins=[0, 12, 18, 35, 60, 100],
    labels=['Child', 'Teen', 'Adult', 'MiddleAge', 'Senior']
)

print("Example of new features:")
print(X_train_eng[['age', 'fare', 'FamilySize', 'IsAlone', 'FarePerPerson', 'AgeGroup']].head())

print("\nUnique Age Groups found:")
print(X_train_eng['AgeGroup'].unique())
```

```
--- Part 2: Creating Features from Domain Knowledge ---
Example of new features:
     age       fare  FamilySize  IsAlone  FarePerPerson   AgeGroup
179    7  74.847623           4        0      18.711906      Child
813   45        NaN           4        0            NaN  MiddleAge
773   40  62.333398           1        1      62.333398  MiddleAge
428   63  48.132391           2        0      24.066196     Senior
592   37  23.567240           4        0       5.891810  MiddleAge

Unique Age Groups found:
['Child', 'MiddleAge', 'Senior', 'Adult', NaN, 'Teen']
Categories (5, object): ['Child' < 'Teen' < 'Adult' < 'MiddleAge' < 'Senior']
```

```python
print("\n--- Part 3: Binning and Discretization ---")

X_train_binned = X_train.copy()
X_train_binned['age'].fillna(X_train_binned['age'].median(), inplace=True)

age_bins = [0, 12, 25, 60, 100]
age_labels = ['Child', 'Young Adult', 'Adult', 'Senior']
X_train_binned['AgeGroup'] = pd.cut(
    X_train_binned['age'],
    bins=age_bins,
    labels=age_labels,
    right=False
)

fare_bins = [0, 50, 100, 200, 500]
fare_labels = ['Low', 'Medium', 'High', 'Very High']
X_train_binned['FareGroup'] = pd.cut(
    X_train_binned['fare'],
    bins=fare_bins,
    labels=fare_labels,
    right=False
)

print("Example of 'AgeGroup' and 'FareGroup' features:")
print(X_train_binned[['age', 'AgeGroup', 'fare', 'FareGroup']].head())

print("\nValue counts for AgeGroup:")
print(X_train_binned['AgeGroup'].value_counts())

print("\nValue counts for FareGroup:")
print(X_train_binned['FareGroup'].value_counts())
```

```
--- Part 3: Binning and Discretization ---
Example of 'AgeGroup' and 'FareGroup' features:
     age AgeGroup        fare FareGroup
179    7    Child  74.847623    Medium
813   45    Adult        NaN       NaN
773   40    Adult  62.333398    Medium
428   63   Senior  48.132391       Low
592   37    Adult  23.567240       Low

Value counts for AgeGroup:
AgeGroup
Adult          464
Young Adult    213
Child           57
Senior          17
Name: count, dtype: int64

Value counts for FareGroup:
FareGroup
Low          341
Medium       210
High         153
Very High     57
Name: count, dtype: int64
```

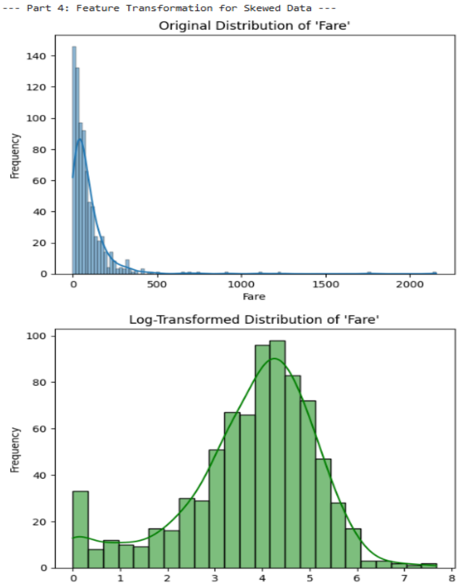
```python
print("\n--- Part 4: Feature Transformation for Skewed Data ---")

sns.histplot(X_train['fare'], kde=True)
plt.title("Original Distribution of 'Fare'")
plt.xlabel("Fare")
plt.ylabel("Frequency")
plt.show()

fare_transformed = np.log1p(X_train['fare'].fillna(0))

sns.histplot(fare_transformed, kde=True, color='green')
plt.title("Log-Transformed Distribution of 'Fare'")
plt.xlabel("Log(Fare + 1)")
plt.ylabel("Frequency")
plt.show()
```

```
--- Part 4: Feature Transformation for Skewed Data ---
```



```python
def create_custom_features(df):
    df_copy = df.copy()

    df_copy['FamilySize'] = df_copy['sibsp'] + df_copy['parch'] + 1
    df_copy['IsAlone'] = (df_copy['FamilySize'] == 1).astype(int)

    df_copy['FarePerPerson'] = df_copy['fare'] / df_copy['FamilySize']

    df_copy['AgeGroup'] = pd.cut(
        df_copy['age'],
        bins=[0, 12, 25, 60, 100],
        labels=['Child', 'Young Adult', 'Adult', 'Senior'],
        right=False
    )

    return df_copy

custom_feature_transformer = FunctionTransformer(create_custom_features, validate=False)

preprocessor_standard = ColumnTransformer(transformers=[

    ('num_impute_scale', Pipeline([
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler())
    ]), ['age', 'fare', 'sibsp', 'parch', 'FamilySize', 'IsAlone', 'FarePerPerson']),

    ('fare_log_impute_scale', Pipeline([
        ('imputer', SimpleImputer(strategy='median')),
        ('log', FunctionTransformer(np.log1p, validate=False)),
        ('scaler', StandardScaler())
    ]), ['fare']),

    ('cat_onehot', OneHotEncoder(handle_unknown='ignore'), ['gender', 'embarkedport', 'travelclass', 'AgeGroup'])
], remainder='drop')

full_pipeline = Pipeline(steps=[
    ('custom_features', custom_feature_transformer),
    ('preprocessor', preprocessor_standard),
    ('classifier', LogisticRegression(random_state=42, max_iter=1000))
])

full_pipeline.fit(X_train, y_train)
y_pred_full = full_pipeline.predict(X_test)
full_accuracy = accuracy_score(y_test, y_pred_full)

print(f"\nBaseline Model Accuracy: {baseline_accuracy:.4f}")
print(f"Full Feature-Engineered Model Accuracy: {full_accuracy:.4f}")
```

```
--- Part 5: Integrating Everything into a Full Pipeline ---

Baseline Model Accuracy: 0.3850
Full Feature-Engineered Model Accuracy: 0.6300
```

ANALYSIS AND DISCUSSION:

The dataset was successfully preprocessed using imputation, encoding, and transformation techniques. Automated workflows simplified data handling and improved model efficiency. Feature scaling and categorical encoding enhanced learning performance. Overall, preprocessing ensured cleaner data, reduced bias, and produced more accurate, reliable model predictions.