Experiment No. 2
Implementation of Feature engineering and pre-processing of data for recommendation systems
Date of Performance:
Date of Submission:
Marks:
Sign:

Aim: Implementation of Feature engineering and pre-processing of data for recommendation systems.

Objective: Understanding the feature engineering and feature scaling and applying pre-processing techniques for recommendations.

Theory:

Preprocessing data is an important step in any data analysis task. It is usually the first step in any data mining or machine learning project. The aim of preprocessing is to clean the data, remove any noise and unwanted information, and prepare the data for further analysis. Feature engineering is the process of creating new features from existing data.
This can be done by transforming existing features, or by combining multiple features to create a new one. Feature engineering is a powerful tool that can help to improve the accuracy of machine learning models. In this guide, we will take a look at some of the most common preprocessing and feature engineering techniques.
Data preprocessing is an essential step in any data science or machine learning project. It is the process of cleaning and preparing the data for analysis. This step is important because it can help improve the accuracy of the results and make the data more manageable.

One of the greatest challenges in data analysis is dealing with missing data. When data is missing, it can be difficult to make accurate predictions. Data preprocessing can help deal with missing data by imputing missing values or using a technique called feature engineering. Feature engineering is the process of creating new features from existing data. This can be done by transforming or combining existing features. For example, you could combine two features to create a new feature that is more predictive. Feature engineering can help improve the accuracy of machine learning models by making the data more representative of the real problem.
There are a few common tasks that data preprocessing typically performs:

- Identifying and handling missing values: This step is important in order to avoid bias in your results. Data preprocessing will identify missing values and either remove them or impute them with another value.
- Identifying and handling outliers: Outliers can often be found in data sets and can sometimes be caused by errors in data entry. Data preprocessing will identify outliers and either remove them or transform them so that they are no longer outliers.
- Feature engineering: This is the process of creating new features from existing data. This can be done by combining existing features, or by Transformations such as scaling or normalization.
- Data scaling and normalization: This step is important for many machine learning algorithms. Some algorithms require that the data be scaled or normalized in order to work correctly. Data preprocessing can take care of this for you.

Removing invalid values

Invalid values are values that are not valid for the data set. Invalid values can occur for a variety of reasons, such as errors in data entry or data that has been corrupted. Invalid values can cause problems with data modeling, so it is important to remove them from the data set.

Imputing missing values

Missing values are values that are not present in the data set. Missing values can occur for a variety of reasons, such as data that has been censored or data that has not been collected. Missing values can cause problems with data modeling, so it is important to impute them.

Scaling data

Scaling data is a method of normalizing data. Normalizing data is important for data modeling because it can help improve the accuracy of models. There are a variety of ways to scale data, but a simple method is to use the “scale()” function in Python. This function will scale the data so that the mean is 0 and the standard deviation is 1.

Implementation:

Frequent Count:

```
import pandas as pd
df = pd.read_csv('titanic.csv')
df.head()
df.isna().sum()
most_occurred = df.Age.value_counts().index[0]
import numpy as np
```

```
def most_frequency_value(dataset, column_name, occurred_variable):
    dataset[column_name+'imputed'] = np.where(dataset[column_name].isna(),
occurred_variable, dataset[column_name])
most_frequency_value(df, 'Age', most_occurred)
df.head()
df.isna().sum()
from sklearn.impute import SimpleImputer
simple_imputer = SimpleImputer(strategy = 'most_frequent')
age_imputed_si = simple_imputer.fit_transform(df[['Age']])
df['age_imputed_si'] = age_imputed_si
df.Age.isna()
df.loc[888]
df.isna().sum()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Ageimputed	age_imputed_si
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	22.0	22.0
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	38.0	38.0
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	26.0	26.0
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	35.0	35.0
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	35.0	35.0

Imputation:

```
import pandas as pd
df = pd.read_csv('titanic.csv')
df.head()
df.isna().sum()
df.shape
def mean_imputation(dataset, column, mean):
    dataset[column+'_mean'] = dataset[column].fillna(mean)
mean = df.Age.mean()
mean_imputation(df, 'Age', mean)
df[['Age', 'Age_mean']].isna()
df.loc[888]
def median_imputation(dataset, column, median):
    dataset[column+'_median'] = dataset[column].fillna(median)
median = df.Age.median()
```

```
median_imputation(df, 'Age', median)
df.loc[888]
df2 = pd.read_csv('titanic.csv')
from sklearn.impute import SimpleImputer
impute_mean = SimpleImputer(strategy = 'mean')
impute_mean.fit(df2[['Age']])
df2['Age_mean'] = impute_mean.transform(df2[['Age']])
df2.loc[888]
impute_median = SimpleImputer(strategy = 'median')
impute_median.fit(df2[['Age']])
df2[['Age_median']] = impute_median.transform(df2[['Age']])
df2.loc[888]
from sklearn.model_selection import train_test_split
X = df2[['Age_mean', 'Pclass']]
y = df2['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=100)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
model.score(X_train, y_train)
X_data = df2[['Age_median', 'Pclass']]
X_train, X_test, y_train, y_test = train_test_split(X_data, y, test_size=0.2,
random_state=100)
model.fit(X_train, y_train)
model.score(X_train, y_train)
```

Output: `0.6980337078651685`

Label Encoding:

```
import pandas as pd
df = pd.read_csv('titanic.csv')
df.head()
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
label_encoding_for_sex = label_encoder.fit_transform(df.Sex)
label_encoder.classes_
df['Sex_Encoded'] = label_encoding_for_sex
df.Embarked.isna().sum()
df.Embarked.unique()
df.dropna(subset = ['Embarked'], inplace=True)
df.Embarked.unique()
embarked_encoded = label_encoder.fit_transform(df.Embarked)
label_encoder.classes_
df['Embarked_encoded'] = embarked_encoded
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Sex_Encoded	Embarked_encoded
	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	1	2
	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0	0
	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	2
	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	0	2
	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	1	2

One Hot Encoding:

```
import pandas as pd
df = pd.read_csv('titanic.csv')
df.head()
from sklearn.preprocessing import OneHotEncoder
one_hot_encoder = OneHotEncoder(sparse = False, dtype = 'int')
df[['female', 'male']] = one_hot_encoder.fit_transform(df[['Sex']])
df = df.drop(columns=['female', 'male'])
df[['female', 'male']] = pd.get_dummies(df.Sex)
df.isna().sum()
df.dropna(subset = ['Embarked'], inplace = True)
df.isna().sum()
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
embarked_encoded = label.fit_transform(df[['Embarked']])
df['Embarked_Encoded'] = embarked_encoded
df = df.drop(columns = ['Embarked'])
from sklearn.impute import SimpleImputer
si = SimpleImputer(strategy = 'mean')
age_impute = si.fit_transform(df[['Age']])
df['Age_Imputed'] = age_impute
df.head()
df.isna().sum()
X = df[['Pclass', 'Age_Imputed', 'Embarked_Encoded', 'female', 'male']]
y = df['Survived']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=100)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
model.score(X_train, y_train)
```

```
y_pred = model.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_pred, y_test)
```

Output: `0.797752808988764`

Conclusion:

In conclusion, effective feature engineering and pre-processing are vital for recommendation systems, enhancing model performance and user experience. Techniques like normalization, encoding, and feature creation optimize data for better insights and predictive accuracy. Robust preprocessing ensures relevant and personalized recommendations, driving user engagement and satisfaction.