Data Preprocessing in Machine Learning using Python

Introduction

Data preprocessing is a crucial step in the machine learning pipeline. It involves transforming raw data into a clean and usable format, which enhances the performance and accuracy of machine learning models. The preprocessing steps vary depending on the type of data and the specific requirements of the model.

Importing and Getting basic insights from data :

https://colab.research.google.com/drive/18Da52qBDuBBsTt1aWeV1cRz5xzuw9Yqy?usp=sharing

How to handle Missing values :

Missing Value : It occurs when no data value is stored for a feature

Reasons - User forgot to fill it , programming error or data was lost while transferring data manually from legacy database.

How to deal with missing data?

DROP DATA

a. drop the whole row

b. drop the whole column

REPLACE DATA

a. replace it by mean

b. replace it by frequency

Code Snippets:

Pandas:

```
[ ] mean_norm_loss=df["normalized-losses"].astype("float").mean()
    mean_norm_loss

    122.0

[ ] df["normalized-losses"].replace(np.nan,mean_norm_loss,inplace=True)
    df['normalized-losses'].head()

    0    122.0
    1    122.0
    2    122.0
    3      164
    4      164
    Name: normalized-losses, dtype: object
```

Scikit.Imputer :

```
[15] imputer_mean=SimpleImputer(missing_values=np.nan,strategy="mean")
     df[['normalized-losses','peak-rpm','bore','stroke','horsepower']]=imputer_mean.fit_transform(df[['normalized-losses','peak-rpm','bore','stroke','horsepower']]).astype("object")
```

Data Formatting

When we have dealt with the null values 90% of data type problems are fixed if they still sustain, then we will do data formatting.

```
ls=['bore','stroke','peak-rpm',"price",'normalized-losses']
```

```
[32] #Convert ["bore", "stroke","peak-rpm","price","normalized-losses"] data types to float data types.
     df[ls]=df[ls].astype(float)
```

Data Bining

Binning is a process of transforming continuous numerical variables into discrete categorical 'bins', for grouped analysis.

```
# Create bins.
bins=np.linspace(min(df['horsepower']),max(df['horsepower']),4)
bins
```

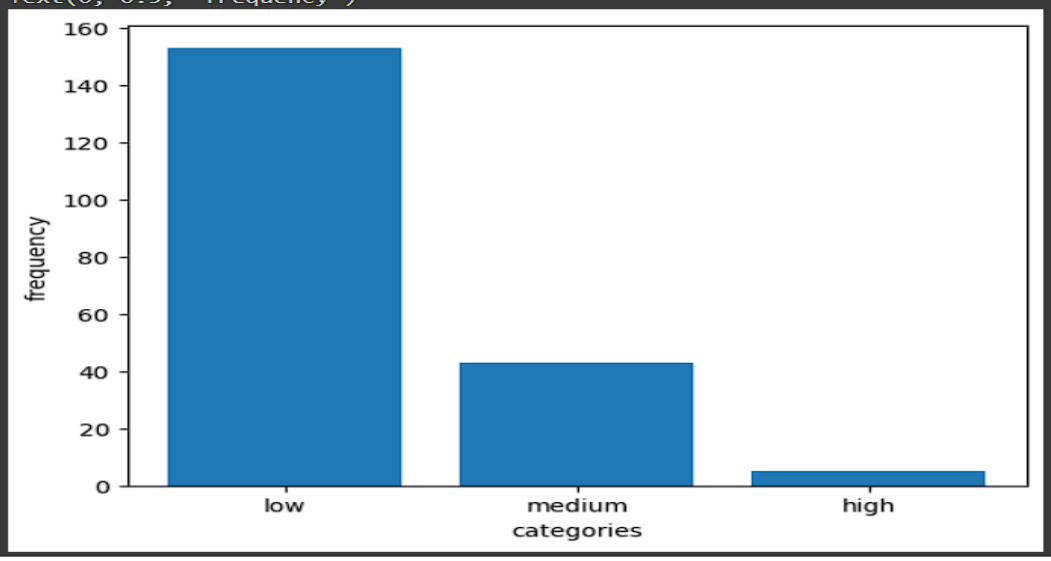
```
array([ 48.        , 119.33333333, 190.66666667, 262.        ])
```

```
[42] # Create group names.
     group_names=['low','medium','high']
```

```
[43] # perform binning.
     df['horsepower_binned']=pd.cut(df['horsepower'],bins,labels=group_names,include_lowest=True)
```

```
[50] # Lets visualize again with bins.
     plt.bar(group_names,df['horsepower_binned'].value_counts())
     plt.xlabel("categories")
     plt.ylabel("frequency")
```

```
Text(0, 0.5, 'frequency')
```



How to Handle Categorical values

Categorical data refers to variables that contain label values rather than numeric values. These labels represent different categories or groups, and the data can be divided into distinct, non-overlapping categories. Categorical data can be either nominal, where the order of the categories does not matter, or ordinal, where the categories have a meaningful order.

Examples of Categorical Data

1. **Nominal Data**: Categories without a natural order.
    - Examples: Gender (Male, Female), Blood Type (A, B, AB, O), Marital Status (Single, Married, Divorced)
2. **Ordinal Data**: Categories with a meaningful order, but the intervals between the categories are not necessarily equal.
    - Examples: Education Level (High School, Bachelor's, Master's, Ph.D.), Customer Satisfaction (Very Unsatisfied, Unsatisfied, Neutral, Satisfied, Very Satisfied)

How to deal with it ?

1)ONE-HOT ENCODING

One-hot encoding transforms each category value into a new categorical column and assigns a binary value (1 or 0) to each column. It is useful when there is no ordinal relationship between the categories.

Code Snippets:

1) Through Pandas

```
# Get dummies with pandas in any variable
country_dummy=pd.get_dummies(df['country'])
country_dummy
```

```
# concatenate the dataframes into original dataframes.
df=pd.concat([df,country_dummy],axis=1)
df
```

```
df.drop('country',axis=1,inplace=True)
df
```

2) Through Scikit-Learn

```
[33] # Lets perform one-hot encoding on country column.
     from sklearn.compose import ColumnTransformer
     from sklearn.preprocessing import OneHotEncoder

[34] ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0])],remainder='passthrough')
     df=pd.DataFrame(ct.fit_transform(df))
     df
```

2) Label Encoding

Label encoding assigns a unique integer to each category value. It is useful when the categorical data has an ordinal relationship.

```
# Lets first perform label encoding
# Labelencoding will be performed on Purchased column as it has only two unique value in it.
df=pd.read_csv("Salary_Dataset.csv")
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Purchased']=le.fit_transform(df['Purchased'])
df
```