Data Preprocessing

Tuesday, March 29, 2022 9:06 PM

1st One

Import:
Numpy
Pandas
Matplotlib.pyplot

Next Step

1. Importing the dataset
2. Taking care of missing data

=> For missing data we import SimpleImputer from sklearn.impute

Note:

- SimpleImputer is a class

- SimpleImputer() [With Paranthesis is object]

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean') #"SimpleImputer()" insid
e paranthesis is an object #.nan means missing value and "strategy='mean'" helps to
perform the mean operation on the dataset
```

- imputer.fit(X[:, 1:3]) #(row, columns) format and .fit is used in order to implement the class in object and the class SimpleImputer and x[:,1:3] here means taking all the rows from the dataset and excluding the first column and the 3rd index value i.e. "Purchased" Column.

- X[:, 1:3] = imputer.transform(X[:, 1:3])
  # "X[:, 1:3]" has been assigned to replace the old blank value with the new one!!!! "imputer.transform(X[:, 1:3])" Adds values to the missing salary and age

3. Encoding the Independent Variable

```
from sklearn.compose import ColumnTransformer # ColumnTransformer is a class
from sklearn.preprocessing import OneHotEncoder # OneHotEncoder is a class which helps
to convert string values to matrix format
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='pass
through')
#encoder helps to transform data into numerical value and "[0]" here means country colum
n whereas in REMAINDER we specify which value shouldn't be transformed
#remainder = "passthrough" helps to keep the column that won't be applied some transform
ation or won't be OneHotEncoded i.e. Age & Salary... TO preserve matrix of feature
X = np.array(ct.fit_transform(X))
#In order to train ML model 'train' function will expect it to be in numpy(the matrix of
 feature).
#So we force the output of this transfomration to be a numpy array
```

4. Encoding the Dependent Variable

```
from sklearn.preprocessing import LabelEncoder #LabelEncoder is a class which helps to c
hange the String value to 0 or 1
le = LabelEncoder()
Y = le.fit_transform(Y)
```

5. Splitting the dataset into Training set and Test set

We will create 4 Separate set:-

i) X-Train
ii) X-Test
iii) Y-Train
iv) Y-Test

```
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 = 1)
# We used "random_state = 1" to get same split
#We need more observation in training set so 80% will be enough for training set and 20% will
be for test set as a result "test_size = 0.2" has been written
```

6. Feature Scaling should be done at the last in order to prevent information leakage for test set.