Name: Arya Sorte

Branch : B.Voc(AIDS)

Division : B

Experiment no.5

Title: Feature Engineering using Python in Machine Learning.

Tools: Anaconda(Jupyter Notebook)

Theory: Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, [1] resulting in an improved model accuracy on unseen data. [2] Feature engineering is very important in machine learning because it can significantly impact the performance of a model.

There are various techniques that can be used in feature engineering to create new features by combining or transforming the existing ones.

1]Filling the missing values using mean ,median,mode imputation

Code:

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
data={
   'Name':['Arya','Kautil','Rehan','Amit','Abhi'],
   'Age':[20, np.nan ,27,19,18],
   'Testscore':[85,90,np.nan,78,92],
   'Grade':['B','A',np.nan,'C','A']
}
df=pd.DataFrame(data)
print("Original Dataset:")
print(df)
num_imputer=SimpleImputer(strategy='mean')
df[['Age','Testscore',]]= num_imputer.fit_transform(df[['Age','Testscore']])
siddhi_imputer=SimpleImputer(strategy='most_frequent')
df[['Grade']]=siddhi_imputer.fit_transform(df[['Grade']])
```

print("\nAfter Mean/Mode Imputation:")

print(df)

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
data={
    'Name':['Arya','Kautil','Rehan','Amit','Abhi'],
    'Age':[20, np.nan ,27,19,18],
    'Testscore':[85,90,np.nan,78,92],
    'Grade':['B','A',np.nan,'C','A']
}
df=pd.DataFrame(data)
print("Original Dataset:")
print(df)

num_imputer=SimpleImputer(strategy='mean')
df[['Age','Testscore',]]= num_imputer.fit_transform(df[['Age','Testscore']])

siddhi_imputer=SimpleImputer(strategy='most_frequent')
df[['Grade']]=siddhi_imputer.fit_transform(df[['Grade']])

print("\nAfter Mean/Mode Imputation:")
print(df)
```

Output:

```
Original Dataset:
     Name   Age  Testscore Grade
0    Arya  20.0       85.0     B
1  Kautil   NaN       90.0     A
2   Rehan  27.0        NaN   NaN
3    Amit  19.0       78.0     C
4    Abhi  18.0       92.0     A

After Mean/Mode Imputation:
     Name   Age  Testscore Grade
0    Arya  20.0      85.00     B
1  Kautil  21.0      90.00     A
2   Rehan  27.0      86.25     A
3    Amit  19.0      78.00     C
4    Abhi  18.0      92.00     A
```

2]Filling the missing values using forward fill and backward fill.

Code:

```
import pandas as pd
import numpy as np
data={
   'Name':['Arya','Kautil','Rehan','Amit','Abhi'],
   'Age':[20, np.nan ,27,19,18],
   'Testscore':[85,90,np.nan,'C','A']
}
df=pd.DataFrame(data)
print("Original Dataset:")
print(df)
df_ffill =df.fillna(method='ffill')
df_bfill =df.fillna(method='bfill')
print("\nAfter Forward Fill (ffill):")
print(df_ffill)
print("\nAfter Backward Fill (bfill):")
print(df_bfill)
```

```python
import pandas as pd
import numpy as np
data={
    'Name':['Arya','Kautil','Rehan','Amit','Abhi'],
    'Age':[20, np.nan ,27,19,18],
    'Testscore':[85,90,np.nan,'C','A']
}
df=pd.DataFrame(data)
print("Original Dataset:")
print(df)
df_ffill =df.fillna(method='ffill')
df_bfill =df.fillna(method='bfill')
print("\nAfter Forward Fill (ffill):")
print(df_ffill)
print("\nAfter Backward Fill (bfill):")
print(df_bfill)
```

Output:

```
Original Dataset:
     Name   Age Testscore
0    Arya  20.0        85
1  Kautil   NaN        90
2   Rehan  27.0       NaN
3    Amit  19.0         C
4    Abhi  18.0         A

After Forward Fill (ffill):
     Name   Age Testscore
0    Arya  20.0        85
1  Kautil  20.0        90
2   Rehan  27.0        90
3    Amit  19.0         C
4    Abhi  18.0         A

After Backward Fill (bfill):
     Name   Age Testscore
0    Arya  20.0        85
1  Kautil  27.0        90
2   Rehan  27.0         C
3    Amit  19.0         C
4    Abhi  18.0         A
```

3]One Hot Encoding.

Code:

import pandas as pd

data={'fruit':['apple','orange','banana','apple']}

df=pd.DataFrame(data)

encoded_df=pd.get_dummies(df, columns=['fruit'])

print(encoded_df)

```
import pandas as pd
data={'fruit':['apple','orange','banana','apple']}
df=pd.DataFrame(data)
encoded_df=pd.get_dummies(df, columns=['fruit'])
print(encoded_df)
```

Output:

```
   fruit_apple  fruit_banana  fruit_orange
0         True         False         False
1        False         False          True
2        False          True         False
3         True         False         False
```

4]Label Encoding:

import pandas as pd

from sklearn.preprocessing import LabelEncoder

df=pd.DataFrame({

'color':['red','blue','green','blue','red'],

'brand':['ford','toyota','ford','bmw','mercedes'],

'type':['SUV','seden','SUV','seden','SUV']

})

let=LabelEncoder()

df['color']=let.fit_transform(df['color'])

df['brand']=let.fit_transform(df['brand'])

df['type']=let.fit_transform(df['type'])

print(df)

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
df=pd.DataFrame({
    'color':['red','blue','green','blue','red'],
    'brand':['ford','toyota','ford','bmw','mercedes'],
    'type':['SUV','seden','SUV','seden','SUV']
})
let=LabelEncoder()
df['color']=let.fit_transform(df['color'])
df['brand']=let.fit_transform(df['brand'])
df['type']=let.fit_transform(df['type'])
print(df)
```

Output:

```
   color  brand  type
0      2      1     0
1      0      3     1
2      1      1     0
3      0      0     1
4      2      2     0
```

Conclusion: Feature engineering is an indispensable part of the machine learning workflow. It's not just about throwing more data at your model; it's about crafting the *right* data representations that empower your model to learn effectively. As we've seen, Python provides a rich ecosystem of tools (pandas, scikit-learn, numpy) to facilitate a wide range of feature engineering techniques.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				