```
import pandas as pd
import numpy as np
```

```
data= {'Name':['Anna','Bob','Raj','Simran','Dev'],
       'Age':[None,32,45,34,29],
       'Gender':['M','M','M','F','M'],
       'job':['Programmer','Cook','Writer','Teacher','Cook']}
```

```
df=pd.DataFrame(data)
df
```

	Name	Age	Gender	job
0	Anna	NaN	M	Programmer
1	Bob	32.0	M	Cook
2	Raj	45.0	M	Writer
3	Simran	34.0	F	Teacher
4	Dev	29.0	M	Cook

```
# Preprocessing Pipeline:
# Drope Name feature
# Impute Age
# Turn Gender into binary/numeric
# one hot encode jobs
```

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder

# Drop Name feature
df=df.drop(['Name'],axis=1)
df
```

	Age	Gender	job
0	NaN	M	Programmer
1	32.0	M	Cook
2	45.0	M	Writer
3	34.0	F	Teacher
4	29.0	M	Cook

```
imputer=SimpleImputer(strategy='mean')
df['Age']= imputer.fit_transform(df[['Age']])
df
#df['Age']=df['Age'].fillna(df['Age'].mean()) use can use fillna also.
```

	Age	Gender	job
0	35.0	M	Programmer
1	32.0	M	Cook
2	45.0	M	Writer
3	34.0	F	Teacher
4	29.0	M	Cook

```
#Turn Gender into Numeric Feature
gender_dct={'M':0,'F':1}
df['Gender']=[gender_dct[g] for g in df ['Gender']]
df
#df['Gender']=df['Gender'].replace({'M':0 ,'F':1}) we can use replace function
```

	Age	Gender	job
0	35.0	0	Programmer
1	32.0	0	Cook
2	45.0	0	Writer
3	34.0	1	Teacher
4	29.0	0	Cook

```
#df1=pd.get_dummies(df)
#df1 we can use get_dummies also,but its add column name to new column+ feature
```

	Age	Gender	job_Cook	job_Programmer	job_Teacher	job_Writer
0	35.0	0	0	1	0	0
1	32.0	0	1	0	0	0
2	45.0	0	0	0	0	1
3	34.0	1	0	0	1	0
4	29.0	0	1	0	0	0

```
#one hot encode jobs

# Create a OneHotEncoder instance
```

```
encoder=OneHotEncoder()

# Transform the 'job' column and convert it to a NumPy array
matrix= encoder.fit_transform(df[['job']]).toarray()

# Get unique job categories and sort them for column names, it'll not create same c
column_names= sorted([i for i in df['job'].unique()])

# Iterate over the transposed matrix and create new columns in the DataFrame
for i in range(len(matrix.T)) :
    df[column_names[i]] = matrix.T[i]

# Drop the original 'job' column as it's no longer needed after encoding
df=df.drop(['job'],axis=1)
```

```
df
```

	Age	Gender	Cook	Programmer	Teacher	Writer
0	35.0	0	0.0	1.0	0.0	0.0
1	32.0	0	1.0	0.0	0.0	0.0
2	45.0	0	0.0	0.0	0.0	1.0
3	34.0	1	0.0	0.0	1.0	0.0
4	29.0	0	1.0	0.0	0.0	0.0

```
#after getting all the desired result lets create class for each feature
#using sklearn.base

from sklearn.base import BaseEstimator, TransformerMixin
class NameDropper(BaseEstimator,TransformerMixin):
    def fit (self,X,y=None):
        return self
    def transform(self,X):
        return X.drop(['Name'],axis=1)
class AgeImputer(BaseEstimator,TransformerMixin):
    def fit (self,X,y=None):
        return self
    def transform(self,X):
        imputer = SimpleImputer(strategy='mean')
        X['Age']= imputer.fit_transform(X[['Age']])
        return X
class FeatureEncoder (BaseEstimator,TransformerMixin):
    def fit (self,X,y=None):
        return self
    def transform(self,X):
        gender_dct={'M':0,'F':1}
        X['Gender']=[gender_dct[g] for g in X ['Gender']]

        encoder=OneHotEncoder()
        matrix= encoder.fit_transform(X[['job']]).toarray()

        column_names= sorted([i for i in X['job'].unique()])

        for i in range(len(matrix.T)) :
            X[column_names[i]] = matrix.T[i]

        return X.drop(['job'],axis=1)


#Creating new dataset for testing
data= {'Name':['Ram','Sham','Kajal','Dev','Rani','Parag'],
       'Age':[None,32,None,34,29,22],
       'Gender':['M','M','F','M','F','M'],
       'job':['Data_Analyst','Cook','Singer','Teacher','Devloper','Cook']}
df2=pd.DataFrame(data)
df2
```

	Name	Age	Gender	job
0	Ram	NaN	M	Data_Analyst
1	Sham	32.0	M	Cook
2	Kajal	NaN	F	Singer
3	Dev	34.0	M	Teacher
4	Rani	29.0	F	Devloper
5	Parag	22.0	M	Cook

```
#calling our function and storing it into variable to test ,I use fit_tranform.

dropper=NameDropper()
imp=AgeImputer()
enc= FeatureEncoder()

enc.fit_transform(imp.fit_transform(dropper.fit_transform(df2)))
```

	Age	Gender	Cook	Data_Analyst	Devloper	Singer	Teacher
0	29.25	0	0.0	1.0	0.0	0.0	0.0
1	32.00	0	1.0	0.0	0.0	0.0	0.0
2	29.25	1	0.0	0.0	0.0	1.0	0.0
3	34.00	0	0.0	0.0	0.0	0.0	1.0
4	29.00	1	0.0	0.0	1.0	0.0	0.0
5	22.00	0	1.0	0.0	0.0	0.0	0.0

```
#creating a data preprocessing pipeline
from sklearn.pipeline import Pipeline
pipe=Pipeline([
    #for droping unwanted column 'Name'
    ('dropper',NameDropper()),
    #for imputing missing values in 'Age'
    ('imputer',AgeImputer()),
    #for endoing features
    ('endoder',FeatureEncoder())
])
#fitting and transforming the dataframe using the defined pipeline
pipe.fit_transform(df2)
```

	Age	Gender	Cook	Data_Analyst	Devloper	Singer	Teacher
0	29.25	0	0.0	1.0	0.0	0.0	0.0
1	32.00	0	1.0	0.0	0.0	0.0	0.0
2	29.25	1	0.0	0.0	0.0	1.0	0.0
3	34.00	0	0.0	0.0	0.0	0.0	1.0