```
# data-->separate(categorical,numerical)
# categorical data--> missing values fill--> encode.
# numerical data--> missing value fill --> standardize.
# apply --> ml model
# calculate model's  performance
# constant , approach name, column name
```

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

```
df=pd.read_csv('/content/Attrition - Attrition (1)d.csv')
df.head(2)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Emplo
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	

2 rows × 35 columns

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
x=df.drop(columns=['EducationField'])
y=df['EducationField']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

```
categorical_features=['Attrition','BusinessTravel','Department']
numerical_features=['Age','DistanceFromHome','Education','EmployeeCount','EmployeeNumber','RelationshipSatisfaction','Stanc
```

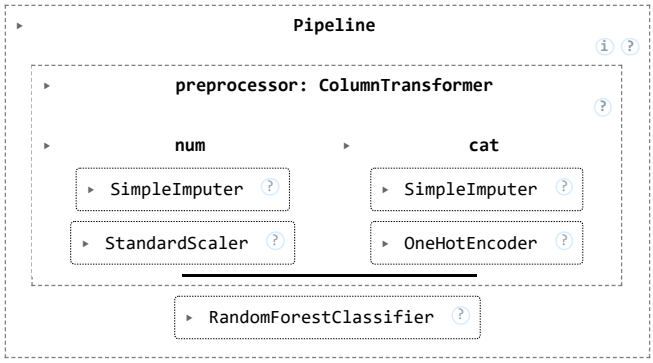
```
numerical_transformer=Pipeline(steps=[
    ('imputer',SimpleImputer(strategy='mean')),
    ('scaler',StandardScaler())
])

categorical_transformer=Pipeline(steps=[
    ('imputer',SimpleImputer(strategy='most_frequent')),
    ('onehot',OneHotEncoder(handle_unknown='ignore'))
])
```

```
## combine transformers
preprocessor=ColumnTransformer(transformers=[
    ('num',numerical_transformer,numerical_features),
    ('cat',categorical_transformer,categorical_features)
])
```

```
## create pipeline for model                                    # randomforestclassifier in algorithm
rf=Pipeline(steps=[
    ('preprocessor',preprocessor),
    ('classifier',RandomForestClassifier())
])
```

```
rf.fit(x_train,y_train)
```




```
y_pred=rf.predict(x_test)
```

```
accuracy_score(y_test,y_pred)
```

```
0.4217687074829932
```