Feature Engineering - Part : C

Feature engineering is the process of transforming raw data into informative and predictive features that can be used effectively by machine learning models.

Different steps used:

1. Imputation.
2. Handling Outliers.
3. Variable Transformation
4. Categorical Encoding.
5. Descritization / Binning.
6. Scalling
7. Feature Creation

Applications of Feature Engineering

Feature engineering is a fundamental step in the data preprocessing pipeline and plays a crucial role in various applications across different domains.

Find me on LinkedIn

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('default')
```

In [ ]:
```
data = {
    'age': np.random.randint(18, 65, 15),
    'gender': ['Male', 'Female', 'Female', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female', 'Male', 'Femal
    'income': np.random.randint(2000, 8000, 15),
    'education': ['High School', 'Bachelor', 'Master', 'PhD', 'High School', 'Bachelor', 'Master', 'PhD', 'Bach
    'occupation': ['Engineer', 'Teacher', 'Doctor', 'Lawyer', 'Engineer', 'Teacher', 'Doctor', 'Lawyer', 'Engin
    'satisfaction': np.random.randint(1, 6, 15),
    'purchase_amount': np.random.randint(100, 1000, 15)
}

data['age'][2] = 0
data['income'][8] = 0
data['occupation'][11] = np.nan

data['purchase_amount'][12] = 5000

dfa = pd.DataFrame(data)

dfa['age'] = dfa['age'].replace(0, np.nan)
dfa['income'] = dfa['income'].replace(0, np.nan)

dfa
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount
0	58.0	Male	2071.0	High School	Engineer	5	900
1	40.0	Female	2438.0	Bachelor	Teacher	5	575
2	NaN	Female	7120.0	Master	Doctor	3	840
3	38.0	Male	4702.0	PhD	Lawyer	4	171
4	46.0	Male	3012.0	High School	Engineer	5	847
5	62.0	Female	7018.0	Bachelor	Teacher	3	795
6	37.0	Male	7192.0	Master	Doctor	1	224
7	26.0	Female	2872.0	PhD	Lawyer	4	105
8	64.0	Female	NaN	Bachelor	Engineer	5	497
9	38.0	Male	7331.0	Master	Teacher	1	133
10	57.0	Female	4349.0	PhD	Doctor	1	969
11	21.0	Male	4406.0	High School	NaN	1	514
12	36.0	Female	2861.0	Bachelor	Engineer	2	5000
13	36.0	Male	3591.0	Master	Teacher	2	319
14	27.0	Female	6239.0	PhD	Doctor	2	493

1. Imputation (Dealing with missing values)

In [ ]:

```
threshold = 0.7

##keeping those columns with missing values rate is lower than threshold
dfa = dfa[dfa.columns[dfa.isna().mean() < threshold]]

dfa
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount
0	58.0	Male	2071.0	High School	Engineer	5	900
1	40.0	Female	2438.0	Bachelor	Teacher	5	575
2	NaN	Female	7120.0	Master	Doctor	3	840
3	38.0	Male	4702.0	PhD	Lawyer	4	171
4	46.0	Male	3012.0	High School	Engineer	5	847
5	62.0	Female	7018.0	Bachelor	Teacher	3	795
6	37.0	Male	7192.0	Master	Doctor	1	224
7	26.0	Female	2872.0	PhD	Lawyer	4	105
8	64.0	Female	NaN	Bachelor	Engineer	5	497
9	38.0	Male	7331.0	Master	Teacher	1	133
10	57.0	Female	4349.0	PhD	Doctor	1	969
11	21.0	Male	4406.0	High School	NaN	1	514
12	36.0	Female	2861.0	Bachelor	Engineer	2	5000
13	36.0	Male	3591.0	Master	Teacher	2	319
14	27.0	Female	6239.0	PhD	Doctor	2	493

In [ ]:

```
##Checking that which attribute contains how many missing values
dfa.isna().sum()
```

Out[ ]:

```
age                1
gender             0
income             1
education          0
occupation         1
satisfaction       0
purchase_amount    0
dtype: int64
```

In [ ]:

```
##Checking the datatypes of the attributes
dfa.dtypes
```

Out[ ]:
```
age                 float64
gender               object
income              float64
education            object
occupation           object
satisfaction          int32
purchase_amount       int32
dtype: object
```

In [ ]:
```
#numerical impuation
dfa["age"] = dfa["age"].fillna(dfa["age"].mean())
dfa["income"] = dfa["income"].fillna(dfa["income"].mean())

#categorical imputation
dfa["occupation"] = dfa["occupation"].fillna(dfa["occupation"].mode()[0])
##mode() represents the most frequent occuring value, and [0] selected 1st value in series

dfa.isna().sum()
```

Out[ ]:
```
age                0
gender             0
income             0
education          0
occupation         0
satisfaction       0
purchase_amount    0
dtype: int64
```

2. Handling outliers

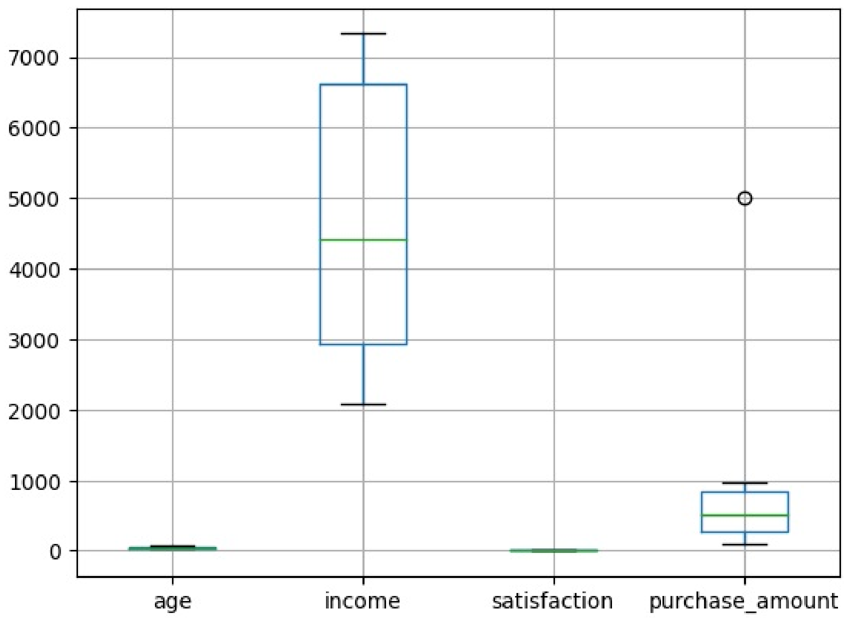
In [ ]:
```
##selecting those columns only whose datatype is integer or a float

numeric_columns = dfa.select_dtypes(include=['int', 'float'])

numeric_columns.boxplot()

##so only purchase_amount attribute contains the outlier
```

Out[ ]: <Axes: >



In [ ]:
```
Q1 = dfa["purchase_amount"].quantile(0.25)
Q3 = dfa["purchase_amount"].quantile(0.75)

IQR = Q3 - Q1

lower_extreme = Q1-1.5*IQR
upper_extreme = Q3+1.5*IQR
```

In [ ]:
```
##detecting outliers
outliers = dfa[(dfa['purchase_amount'] < lower_extreme) | (dfa['purchase_amount'] > upper_extreme)]
outliers
```

Out[ ]:

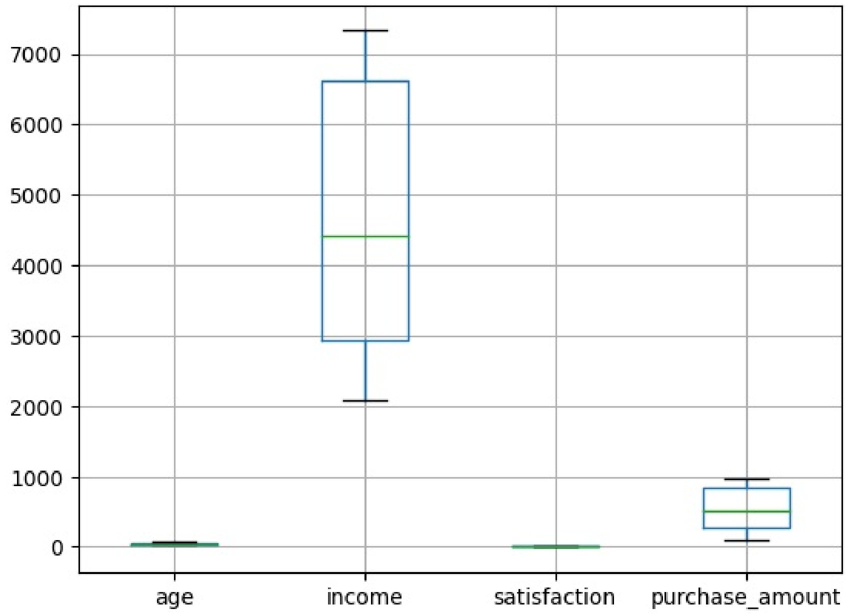
	age	gender	income	education	occupation	satisfaction	purchase_amount
12	36.0	Female	2861.0	Bachelor	Engineer	2	5000

In [ ]:
```
##replacing oulier with mean
```

```
##replacing outlier with mean
dfa.loc[dfa['purchase_amount'] < lower_extreme, "purchase_amount"] = dfa["purchase_amount"].mean()
dfa.loc[dfa['purchase_amount'] > upper_extreme, "purchase_amount"] = dfa["purchase_amount"].mean()

dfa.boxplot()
```

Out[ ]: <Axes: >



3. Variable transformation

```
def convert(s):
    dfa[s] = dfa[s].astype('int64')

fl = list(dfa.select_dtypes('float'))   #selecting all those attribute with float datatype

for k in fl:
    convert(k)

dfa
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount
0	58	Male	2071	High School	Engineer	5	900
1	40	Female	2438	Bachelor	Teacher	5	575
2	41	Female	7120	Master	Doctor	3	840
3	38	Male	4702	PhD	Lawyer	4	171
4	46	Male	3012	High School	Engineer	5	847
5	62	Female	7018	Bachelor	Teacher	3	795
6	37	Male	7192	Master	Doctor	1	224
7	26	Female	2872	PhD	Lawyer	4	105
8	64	Female	4657	Bachelor	Engineer	5	497
9	38	Male	7331	Master	Teacher	1	133
10	57	Female	4349	PhD	Doctor	1	969
11	21	Male	4406	High School	Doctor	1	514
12	36	Female	2861	Bachelor	Engineer	2	825
13	36	Male	3591	Master	Teacher	2	319
14	27	Female	6239	PhD	Doctor	2	493

```
dfa["income_log"] = np.log(dfa["income"])
dfa["purchase_amount_scale"] = (dfa["purchase_amount"] - dfa["purchase_amount"].mean())/dfa["purchase_amount"].s
dfa
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount	income_log	purchase_amount_scale
0	58	Male	2071	High School	Engineer	5	900	7.635787	1.162885
1	40	Female	2438	Bachelor	Teacher	5	575	7.798933	0.091836
2	41	Female	7120	Master	Doctor	3	840	8.870663	0.965153
3	38	Male	4702	PhD	Lawyer	4	171	8.455743	-1.239561
4	46	Male	3012	High School	Engineer	5	847	8.010360	0.988222
5	62	Female	7018	Bachelor	Teacher	3	795	8.856234	0.816854
6	37	Male	7192	Master	Doctor	1	224	8.880725	-1.064898
7	26	Female	2872	PhD	Lawyer	4	105	7.962764	-1.457067
8	64	Female	4657	Bachelor	Engineer	5	497	8.446127	-0.165216
9	38	Male	7331	Master	Teacher	1	133	8.899867	-1.364792
10	57	Female	4349	PhD	Doctor	1	969	8.377701	1.390277
11	21	Male	4406	High School	Doctor	1	514	8.390723	-0.109192
12	36	Female	2861	Bachelor	Engineer	2	825	7.958926	0.915720
13	36	Male	3591	Master	Teacher	2	319	8.186186	-0.751822
14	27	Female	6239	PhD	Doctor	2	493	8.738575	-0.178398

4. Categorical Encoding

In [ ]:
```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, OrdinalEncoder
```

In [ ]:
```
## one-hot encoding
df_one_hot = pd.get_dummies(dfa, columns=['gender', 'education', 'occupation'])
df_one_hot = df_one_hot.replace({True:1, False:0})
dfb = df_one_hot.copy()
dfb
```

Out[ ]:

	age	income	satisfaction	purchase_amount	income_log	purchase_amount_scale	gender_Female	gender_Male	education_Ba
0	58	2071	5	900	7.635787	1.162885	0	1	
1	40	2438	5	575	7.798933	0.091836	1	0	
2	41	7120	3	840	8.870663	0.965153	1	0	
3	38	4702	4	171	8.455743	-1.239561	0	1	
4	46	3012	5	847	8.010360	0.988222	0	1	
5	62	7018	3	795	8.856234	0.816854	1	0	
6	37	7192	1	224	8.880725	-1.064898	0	1	
7	26	2872	4	105	7.962764	-1.457067	1	0	
8	64	4657	5	497	8.446127	-0.165216	1	0	
9	38	7331	1	133	8.899867	-1.364792	0	1	
10	57	4349	1	969	8.377701	1.390277	1	0	
11	21	4406	1	514	8.390723	-0.109192	0	1	
12	36	2861	2	825	7.958926	0.915720	1	0	
13	36	3591	2	319	8.186186	-0.751822	0	1	
14	27	6239	2	493	8.738575	-0.178398	1	0	

In [ ]:
```
## Label encoding
dfa['gender'] = dfa['gender'].map({'Male':1, 'Female':0})
# dfa['gender'] = np.where(dfa['gender']=="Male", 1, 0)
dfa
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount	income_log	purchase_amount_scale
0	58	1	2071	High School	Engineer	5	900	7.635787	1.162885
1	40	0	2438	Bachelor	Teacher	5	575	7.798933	0.091836
2	41	0	7120	Master	Doctor	3	840	8.870663	0.965153
3	38	1	4702	PhD	Lawyer	4	171	8.455743	-1.239561
4	46	1	3012	High School	Engineer	5	847	8.010360	0.988222
5	62	0	7018	Bachelor	Teacher	3	795	8.856234	0.816854
6	37	1	7192	Master	Doctor	1	224	8.880725	-1.064898
7	26	0	2872	PhD	Lawyer	4	105	7.962764	-1.457067
8	64	0	4657	Bachelor	Engineer	5	497	8.446127	-0.165216
9	38	1	7331	Master	Teacher	1	133	8.899867	-1.364792
10	57	0	4349	PhD	Doctor	1	969	8.377701	1.390277
11	21	1	4406	High School	Doctor	1	514	8.390723	-0.109192
12	36	0	2861	Bachelor	Engineer	2	825	7.958926	0.915720
13	36	1	3591	Master	Teacher	2	319	8.186186	-0.751822
14	27	0	6239	PhD	Doctor	2	493	8.738575	-0.178398

5. Descretization or Binning

In [ ]:
```
no_of_bins = 3
bin_labels = ['low', 'medium', 'high']

dfa["income_bin"] = pd.cut(dfa["income"], bins=no_of_bins, labels=bin_labels)

dfa["purchase_amount_bin"] = pd.qcut(dfa["purchase_amount"], q=3, labels=bin_labels)

dfa
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount	income_log	purchase_amount_scale	income_bin
0	58	1	2071	High School	Engineer	5	900	7.635787	1.162885	low
1	40	0	2438	Bachelor	Teacher	5	575	7.798933	0.091836	low
2	41	0	7120	Master	Doctor	3	840	8.870663	0.965153	high
3	38	1	4702	PhD	Lawyer	4	171	8.455743	-1.239561	medium
4	46	1	3012	High School	Engineer	5	847	8.010360	0.988222	low
5	62	0	7018	Bachelor	Teacher	3	795	8.856234	0.816854	high
6	37	1	7192	Master	Doctor	1	224	8.880725	-1.064898	high
7	26	0	2872	PhD	Lawyer	4	105	7.962764	-1.457067	low
8	64	0	4657	Bachelor	Engineer	5	497	8.446127	-0.165216	medium
9	38	1	7331	Master	Teacher	1	133	8.899867	-1.364792	high
10	57	0	4349	PhD	Doctor	1	969	8.377701	1.390277	medium
11	21	1	4406	High School	Doctor	1	514	8.390723	-0.109192	medium
12	36	0	2861	Bachelor	Engineer	2	825	7.958926	0.915720	low
13	36	1	3591	Master	Teacher	2	319	8.186186	-0.751822	low
14	27	0	6239	PhD	Doctor	2	493	8.738575	-0.178398	high

6. Scalling

In [ ]:
```
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
## Standard z-score scalling/normalization

dfc = dfa.copy()

numeric_columns = ['income', 'purchase_amount']

##Initializing the scalar object
```

```
##The StandardScaler scales the data by subtracting the mean and dividing by std.
scaler = StandardScaler()

dfc[numeric_columns] = scaler.fit_transform(dfc[numeric_columns])

dfc
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount	income_log	purchase_amount_scale	income_bi
0	58	1	-1.420177	High School	Engineer	5	1.203700	7.635787	1.162885	lo
1	40	0	-1.218649	Bachelor	Teacher	5	0.095059	7.798933	0.091836	lo
2	41	0	1.352342	Master	Doctor	3	0.999028	8.870663	0.965153	hig
3	38	1	0.024564	PhD	Lawyer	4	-1.283068	8.455743	-1.239561	mediu
4	46	1	-0.903453	High School	Engineer	5	1.022907	8.010360	0.988222	lo
5	62	0	1.296332	Bachelor	Teacher	3	0.845524	8.856234	0.816854	hig
6	37	1	1.391879	Master	Doctor	1	-1.102274	8.880725	-1.064898	hig
7	26	0	-0.980330	PhD	Lawyer	4	-1.508207	7.962764	-1.457067	lo
8	64	0	-0.000146	Bachelor	Engineer	5	-0.171015	8.446127	-0.165216	mediu
9	38	1	1.468207	Master	Teacher	1	-1.412694	8.899867	-1.364792	hig
10	57	0	-0.169276	PhD	Doctor	1	1.439073	8.377701	1.390277	mediu
11	21	1	-0.137976	High School	Doctor	1	-0.113025	8.390723	-0.109192	mediu
12	36	0	-0.986370	Bachelor	Engineer	2	0.947860	7.958926	0.915720	lo
13	36	1	-0.585511	Master	Teacher	2	-0.778209	8.186186	-0.751822	lo
14	27	0	0.868565	PhD	Doctor	2	-0.184660	8.738575	-0.178398	hig

7. Feature Creation

In [ ]:

```
bins_no = [18, 25, 35, 50, 65]
bin_labels = ['18-25', '26-35', '36-50', '51-65']
dfc["age_group"] = pd.cut(dfc["age"], bins = bins_no, labels = bin_labels)
dfc
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount	income_log	purchase_amount_scale	income_bi
0	58	1	-1.420177	High School	Engineer	5	1.203700	7.635787	1.162885	lo
1	40	0	-1.218649	Bachelor	Teacher	5	0.095059	7.798933	0.091836	lo
2	41	0	1.352342	Master	Doctor	3	0.999028	8.870663	0.965153	hig
3	38	1	0.024564	PhD	Lawyer	4	-1.283068	8.455743	-1.239561	mediu
4	46	1	-0.903453	High School	Engineer	5	1.022907	8.010360	0.988222	lo
5	62	0	1.296332	Bachelor	Teacher	3	0.845524	8.856234	0.816854	hig
6	37	1	1.391879	Master	Doctor	1	-1.102274	8.880725	-1.064898	hig
7	26	0	-0.980330	PhD	Lawyer	4	-1.508207	7.962764	-1.457067	lo
8	64	0	-0.000146	Bachelor	Engineer	5	-0.171015	8.446127	-0.165216	mediu
9	38	1	1.468207	Master	Teacher	1	-1.412694	8.899867	-1.364792	hig
10	57	0	-0.169276	PhD	Doctor	1	1.439073	8.377701	1.390277	mediu
11	21	1	-0.137976	High School	Doctor	1	-0.113025	8.390723	-0.109192	mediu
12	36	0	-0.986370	Bachelor	Engineer	2	0.947860	7.958926	0.915720	lo
13	36	1	-0.585511	Master	Teacher	2	-0.778209	8.186186	-0.751822	lo
14	27	0	0.868565	PhD	Doctor	2	-0.184660	8.738575	-0.178398	hig

> Here we have 2 Datasets

> One-hot encoded

In [ ]:
```
dfb
```

Out[ ]:

	age	income	satisfaction	purchase_amount	income_log	purchase_amount_scale	gender_Female	gender_Male	education_Ba
0	58	2071	5	900	7.635787	1.162885	0	1	
1	40	2438	5	575	7.798933	0.091836	1	0	
2	41	7120	3	840	8.870663	0.965153	1	0	
3	38	4702	4	171	8.455743	-1.239561	0	1	
4	46	3012	5	847	8.010360	0.988222	0	1	
5	62	7018	3	795	8.856234	0.816854	1	0	
6	37	7192	1	224	8.880725	-1.064898	0	1	
7	26	2872	4	105	7.962764	-1.457067	1	0	
8	64	4657	5	497	8.446127	-0.165216	1	0	
9	38	7331	1	133	8.899867	-1.364792	0	1	
10	57	4349	1	969	8.377701	1.390277	1	0	
11	21	4406	1	514	8.390723	-0.109192	0	1	
12	36	2861	2	825	7.958926	0.915720	1	0	
13	36	3591	2	319	8.186186	-0.751822	0	1	
14	27	6239	2	493	8.738575	-0.178398	1	0	

> final dataframe

In [ ]:
```
dfc
```

Out[ ]:

	age	gender	income	education	occupation	satisfaction	purchase_amount	income_log	purchase_amount_scale	income_bi
0	58	1	-1.420177	High School	Engineer	5	1.203700	7.635787	1.162885	lo
1	40	0	-1.218649	Bachelor	Teacher	5	0.095059	7.798933	0.091836	lo
2	41	0	1.352342	Master	Doctor	3	0.999028	8.870663	0.965153	hig
3	38	1	0.024564	PhD	Lawyer	4	-1.283068	8.455743	-1.239561	mediu
4	46	1	-0.903453	High School	Engineer	5	1.022907	8.010360	0.988222	lo
5	62	0	1.296332	Bachelor	Teacher	3	0.845524	8.856234	0.816854	hig
6	37	1	1.391879	Master	Doctor	1	-1.102274	8.880725	-1.064898	hig
7	26	0	-0.980330	PhD	Lawyer	4	-1.508207	7.962764	-1.457067	lo
8	64	0	-0.000146	Bachelor	Engineer	5	-0.171015	8.446127	-0.165216	mediu
9	38	1	1.468207	Master	Teacher	1	-1.412694	8.899867	-1.364792	hig
10	57	0	-0.169276	PhD	Doctor	1	1.439073	8.377701	1.390277	mediu
11	21	1	-0.137976	High School	Doctor	1	-0.113025	8.390723	-0.109192	mediu
12	36	0	-0.986370	Bachelor	Engineer	2	0.947860	7.958926	0.915720	lo
13	36	1	-0.585511	Master	Teacher	2	-0.778209	8.186186	-0.751822	lo
14	27	0	0.868565	PhD	Doctor	2	-0.184660	8.738575	-0.178398	hig