EXPERIMENT NO: 5

Feature Scaling for given Dataset

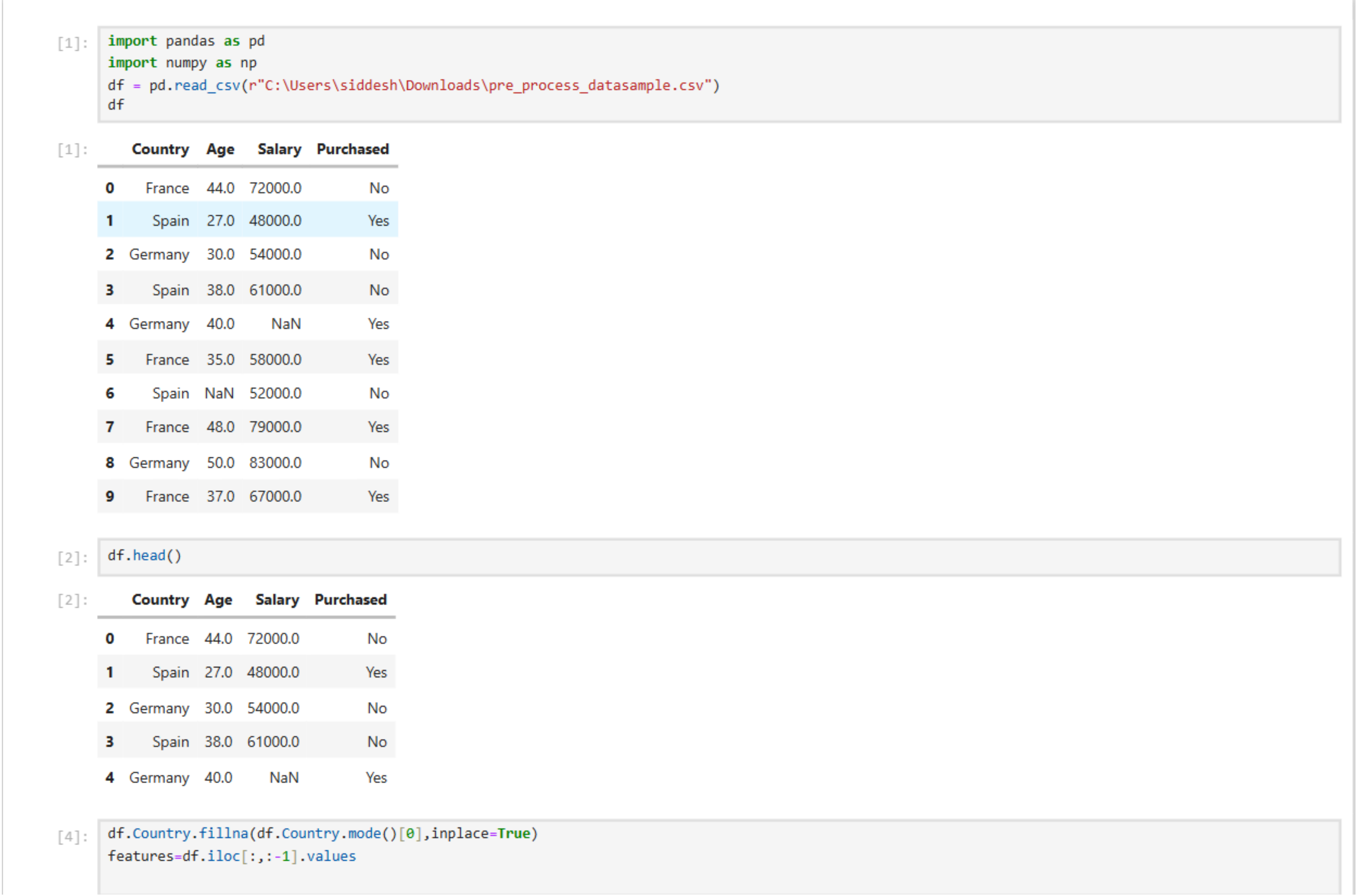
Aim:

To write the Python program to perform and understand the importance of feature scaling.

Algorithm:

1. Load the dataset and inspect its structure and contents.
2. Identify and handle missing values using mean imputation. Encode categorical variables using OneHotEncoder.
3. Normalize numerical features using MinMaxScaler.
4. Standardize features using StandardScaler for uniform scaling.
5. Combine processed features into a final dataset for analysis.

Program:



[1]:
```
import pandas as pd
import numpy as np
df = pd.read_csv(r"C:\Users\siddesh\Downloads\pre_process_datasample.csv")
df
```

[1]:

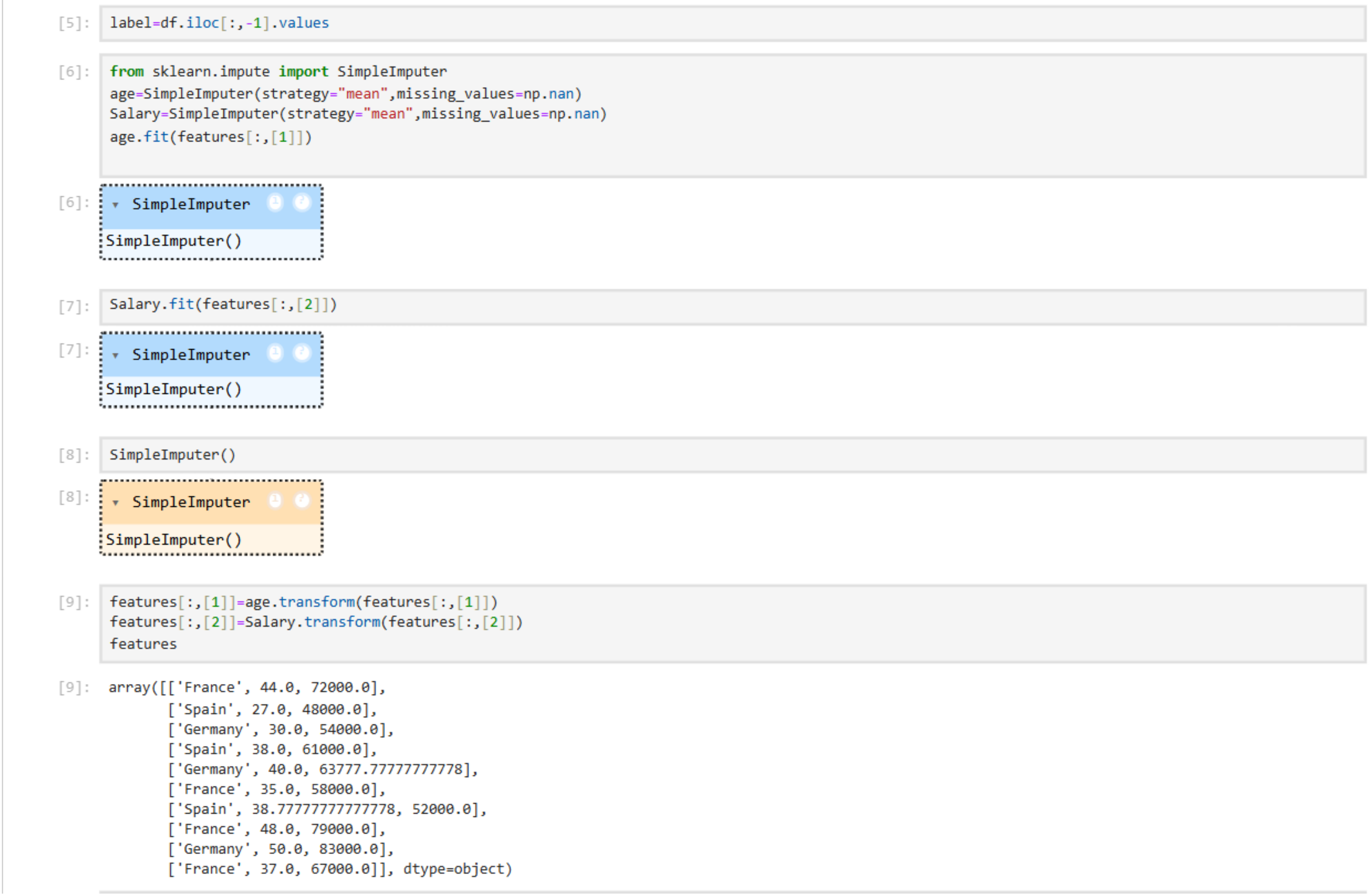
	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

[2]:
```
df.head()
```

[2]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

[4]:
```
df.Country.fillna(df.Country.mode()[0],inplace=True)
features=df.iloc[:,:-1].values
```



[5]:
```
label=df.iloc[:,-1].values
```

[6]:
```
from sklearn.impute import SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np.nan)
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
age.fit(features[:,[1]])
```

[6]:
```
▾ SimpleImputer
SimpleImputer()
```

[7]:
```
Salary.fit(features[:,[2]])
```

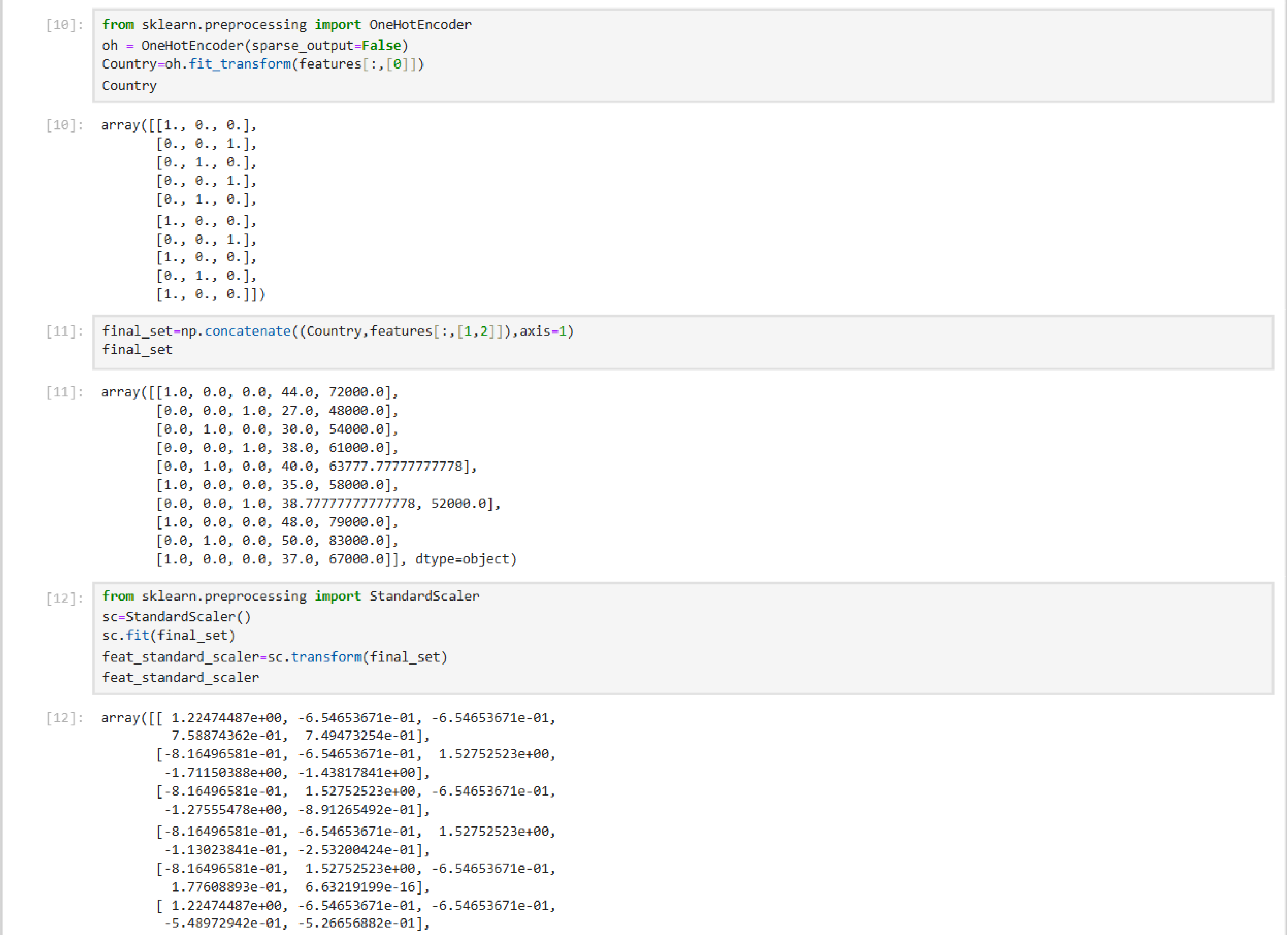
[7]:
```
▾ SimpleImputer
SimpleImputer()
```

[8]:
```
SimpleImputer()
```

[8]:
```
▾ SimpleImputer
SimpleImputer()
```

[9]:
```
features[:,[1]]=age.transform(features[:,[1]])
features[:,[2]]=Salary.transform(features[:,[2]])
features
```

[9]:
```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```



```
[10]: from sklearn.preprocessing import OneHotEncoder
      oh = OneHotEncoder(sparse_output=False)
      Country=oh.fit_transform(features[:,[0]])
      Country
```

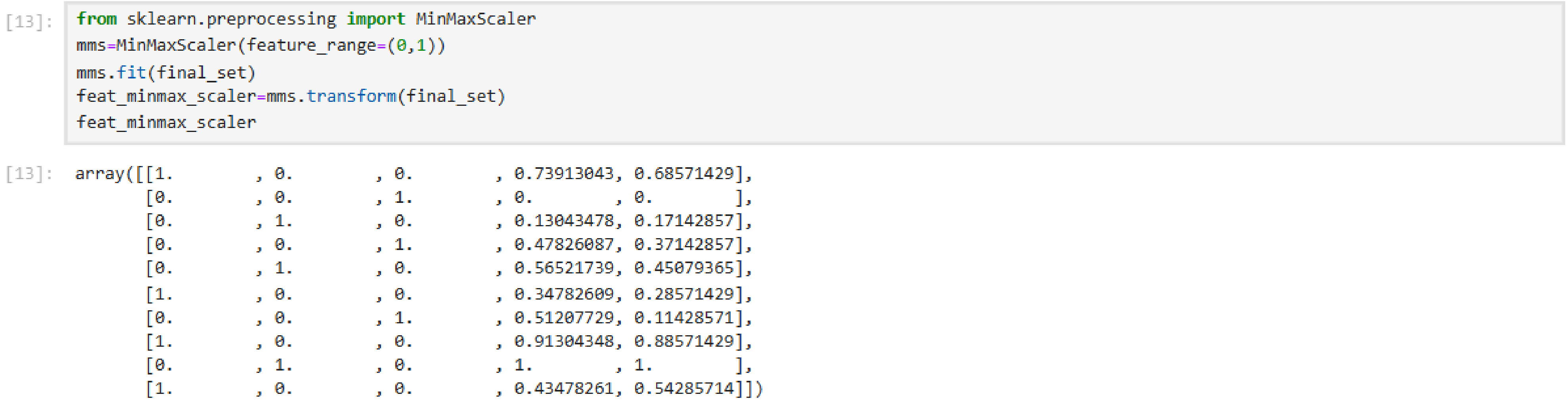
```
[10]: array([[1., 0., 0.],
             [0., 0., 1.],
             [0., 1., 0.],
             [0., 0., 1.],
             [0., 1., 0.],
             [1., 0., 0.],
             [0., 0., 1.],
             [1., 0., 0.],
             [0., 1., 0.],
             [1., 0., 0.]])
```

```
[11]: final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
      final_set
```

```
[11]: array([[1.0, 0.0, 0.0, 44.0, 72000.0],
             [0.0, 0.0, 1.0, 27.0, 48000.0],
             [0.0, 1.0, 0.0, 30.0, 54000.0],
             [0.0, 0.0, 1.0, 38.0, 61000.0],
             [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
             [1.0, 0.0, 0.0, 35.0, 58000.0],
             [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
             [1.0, 0.0, 0.0, 48.0, 79000.0],
             [0.0, 1.0, 0.0, 50.0, 83000.0],
             [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

```
[12]: from sklearn.preprocessing import StandardScaler
      sc=StandardScaler()
      sc.fit(final_set)
      feat_standard_scaler=sc.transform(final_set)
      feat_standard_scaler
```

```
[12]: array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
               7.58874362e-01,  7.49473254e-01],
             [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
              -1.71150388e+00, -1.43817841e+00],
             [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
              -1.27555478e+00, -8.91265492e-01],
             [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
              -1.13023841e-01, -2.53200424e-01],
             [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
               1.77608893e-01,  6.63219199e-16],
             [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
              -5.48972942e-01, -5.26656882e-01],
```



```
[13]: from sklearn.preprocessing import MinMaxScaler
      mms=MinMaxScaler(feature_range=(0,1))
      mms.fit(final_set)
      feat_minmax_scaler=mms.transform(final_set)
      feat_minmax_scaler
```

```
[13]: array([[1.        , 0.        , 0.        , 0.73913043, 0.68571429],
             [0.        , 0.        , 1.        , 0.        , 0.        ],
             [0.        , 1.        , 0.        , 0.13043478, 0.17142857],
             [0.        , 0.        , 1.        , 0.47826087, 0.37142857],
             [0.        , 1.        , 0.        , 0.56521739, 0.45079365],
             [1.        , 0.        , 0.        , 0.34782609, 0.28571429],
             [0.        , 0.        , 1.        , 0.51207729, 0.11428571],
             [1.        , 0.        , 0.        , 0.91304348, 0.88571429],
             [0.        , 1.        , 0.        , 1.        , 1.        ],
             [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```

Result:

Thus, the Python program is executed successfully for feature scaling of the given dataset using Pandas and Scikit-learn techniques.