

```
In [2]: import pandas as pd
        from sdv.timeseries import PAR
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
        from sdv.tabular import GaussianCopula
        from datetime import datetime
        from datetime import timedelta, date
        from datetime import timedelta
        import requests
        import time
        import random
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        from scipy.stats import norm
```

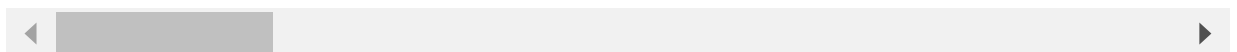
```
In [3]: df = pd.read_excel('RawData.xlsx')
```

```
In [4]: df.head()
```

```
Out[4]:
```

	Year	Month	user_account_id	user_lifetime	user_intake	user_no_outgoing_activity_in_days	user_
0	2013	6	13	1000	0		1
1	2013	6	14	1000	0		25
2	2013	6	18	1005	0		8
3	2013	6	27	1013	0		11
4	2013	6	32	1032	0		2

5 rows × 66 columns



## Method 1: Random Forest classifier Method

```
In [6]: from sklearn.feature_selection import RFE
        from sklearn.ensemble import RandomForestClassifier

        # Load the dataset
        X = df.iloc[:, 0 : 65]
        y = df.iloc[:,65]

        # Create a Random Forest classifier
        model = RandomForestClassifier()

        # Create the RFE object and rank each feature
        rfe = RFE(model, n_features_to_select = 13)
        rfe.fit(X, y)

        # Print the rank of each feature
        print("Feature ranking:")
```

```
for i, feature in enumerate(rfe.support_):  
    print(f"{i+1}: {feature}")
```

Feature ranking:

1: False  
2: False  
3: True  
4: True  
5: False  
6: False  
7: True  
8: True  
9: False  
10: False  
11: False  
12: False  
13: False  
14: False  
15: False  
16: True  
17: False  
18: True  
19: False  
20: True  
21: True  
22: False  
23: False  
24: False  
25: True  
26: False  
27: False  
28: False  
29: True  
30: False  
31: False  
32: False  
33: False  
34: False  
35: False  
36: False  
37: False  
38: False  
39: False  
40: False  
41: False  
42: False  
43: False  
44: False  
45: False  
46: False  
47: True  
48: False  
49: False  
50: False  
51: False  
52: False  
53: False  
54: False  
55: False  
56: False  
57: True  
58: False  
59: True  
60: False

```
61: False
62: False
63: False
64: False
65: False
```

## Method 2: f\_classif method

```
In [48]: from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

x = pd.read_excel("RawData.xlsx")

X = x.drop('Churn', axis=1)
y = x['Churn']

# Select the top 10 features
selector = SelectKBest(f_classif, k=10)
X_new = selector.fit_transform(X, y)

# Get the selected feature names
selected_feature_names = df[X.columns[selector.get_support()]]
list(selected_feature_names.columns)
```

```
C:\Users\gangavarapu.deep\Anaconda3\lib\site-packages\sklearn\feature_selection\_univariate_selection.py:114: UserWarning: Features [0] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx,
C:\Users\gangavarapu.deep\Anaconda3\lib\site-packages\sklearn\feature_selection\_univariate_selection.py:116: RuntimeWarning: invalid value encountered in true_divide
  f = msb / msd
```

```
Out[48]: ['user_lifetime',
'user_has_outgoing_calls',
'calls_outgoing_inactive_days',
'calls_outgoing_to_onnet_inactive_days ',
'calls_outgoing_to_offnet_inactive_days ',
'calls_outgoing_to_abroad_inactive_days ',
'sms_outgoing_inactive_days ',
'sms_outgoing_to_onnet_inactive_days ',
'sms_outgoing_to_offnet_inactive_days ',
'sms_outgoing_to_abroad_inactive_days ']
```

```
In [46]:
```

```
Out[46]: ['user_lifetime',
'user_has_outgoing_calls',
'calls_outgoing_inactive_days',
'calls_outgoing_to_onnet_inactive_days ',
'calls_outgoing_to_offnet_inactive_days ',
'calls_outgoing_to_abroad_inactive_days ',
'sms_outgoing_inactive_days ',
'sms_outgoing_to_onnet_inactive_days ',
'sms_outgoing_to_offnet_inactive_days ',
'sms_outgoing_to_abroad_inactive_days ']
```

## Method 3: Linear Regression

```
In [42]: from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

# X is your dataset with features and y is the target variable
```

```

X = x.drop('Churn', axis=1)
y = x['Churn']

# Use a linear regression model to select the top 10 features
lr = LinearRegression()
rfe = RFE(lr, n_features_to_select=10)
X_new = rfe.fit_transform(X, y)

# Get the selected feature names
selected_feature_names = X.columns[rfe.get_support()]

print(selected_feature_names)

```

```

Index(['Month', 'user_intake', 'user_spendings', 'user_has_outgoing_calls',
      'user_has_outgoing_sms', 'user_use_gprs', 'user_does_reload',
      'calls_outgoing_spendings', 'sms_outgoing_spendings ',
      'gprs_spendings '],
      dtype='object')

```

## Method 4: Filter methods - Correlation Coefficient Method

### a. Threshold value method

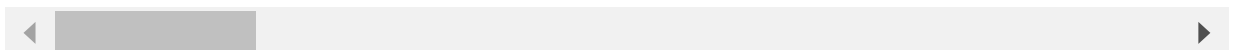
In [33]:

```
df.corr()
```

Out[33]:

	Year	Month	user_account_id	user_lifetime	user_intake
	Year	NaN	NaN	NaN	NaN
	Month	NaN	1.000000	0.537110	0.670438
	user_account_id	NaN	0.537110	1.000000	0.474652
	user_lifetime	NaN	0.670438	1.000000	0.124386
	user_intake	NaN	0.213441	0.254815	1.000000
	...	...	...	...	...
last_100_sms_outgoing_to_onnet_count2	NaN	-0.075731	-0.045914	-0.102034	-0.027395
last_100_sms_outgoing_to_offnet_count	NaN	-0.122698	-0.055299	-0.168767	-0.040466
last_100_sms_outgoing_to_abroad_count	NaN	-0.044493	-0.039027	-0.059954	-0.015998
last_100_gprs_usage	NaN	-0.019019	0.004402	-0.037494	0.008617
Churn	NaN	0.476116	0.357816	0.529719	-0.014725

66 rows × 66 columns



In [27]:

```

def correlation(x, threshold):
    features = dict()
    corr_matrix = x
    feat_size = len(corr_matrix.columns) #Whole Data
    for i in range(feat_size-1):
        if abs(corr_matrix.iloc[i,feat_size-1])>threshold :

```

```

        features.update({corr_matrix.columns[i]:corr_matrix.iloc[i,feat_size-1]})
    return features

```

```

In [29]: corrmat = df.corr()
        sample = correlation(corrmat,0.4)
        sample

```

```

Out[29]: {'Month': 0.4761157054950701,
        'user_lifetime': 0.5297193170811494,
        'user_has_outgoing_calls': -0.5537898650179925,
        'user_has_outgoing_sms': -0.49021418360982316,
        'calls_outgoing_inactive_days': 0.5461726281768687,
        'calls_outgoing_to_onnet_inactive_days ': 0.5461726281768687,
        'calls_outgoing_to_offnet_inactive_days ': 0.5461726281768687,
        'calls_outgoing_to_abroad_inactive_days ': 0.5461726281768687,
        'sms_outgoing_inactive_days ': 0.5402346905312854,
        'sms_outgoing_to_onnet_inactive_days ': 0.5402346905312854,
        'sms_outgoing_to_offnet_inactive_days ': 0.5402346905312854,
        'sms_outgoing_to_abroad_inactive_days ': 0.5402346905312854}

```

## b. "spearmanr" method

```

In [35]: import pandas as pd
        from scipy.stats import spearmanr

        # Load the data
        data = df

        # separate the features and target
        X = data.drop("Churn", axis=1)
        y = data["Churn"]

        # calculate the correlation matrix
        corr_matrix = np.zeros((len(X.columns),1))

        for i in range(len(X.columns)):
            corr_matrix[i] = spearmanr(X[X.columns[i]],y)[0]

        # Create a correlation dataframe
        corr_df = pd.DataFrame({'feature': X.columns, 'corr': corr_matrix.flatten()})

        # sort the dataframe in descending order
        corr_df_positive = corr_df.sort_values('corr',ascending=False)
        corr_df_negative = corr_df.sort_values('corr',ascending=True)

```

C:\Users\gangavarapu.deep\Anaconda3\lib\site-packages\scipy\stats\stats.py:4484: SpearmanRConstantInputWarning: An input array is constant; the correlation coefficient is not defined.

```
warnings.warn(SpearmanRConstantInputWarning())
```

```

In [40]: # Positive corelated features
        corr_df_positive.head()

```

```

Out[40]:

```

	feature	corr
36	sms_outgoing_inactive_days	0.517625
39	sms_outgoing_to_onnet_inactive_days	0.517625
42	sms_outgoing_to_offnet_inactive_days	0.517625

	<b>feature</b>	<b>corr</b>
<b>45</b>	sms_outgoing_to_abroad_inactive_days	0.517625
<b>24</b>	calls_outgoing_to_onnet_inactive_days	0.500037

In [37]:

```
# Negative correlated features
corr_df_negative.head()
```

Out[37]:

	<b>feature</b>	<b>corr</b>
<b>8</b>	user_has_outgoing_calls	-0.553790
<b>56</b>	last_100_calls_outgoing_duration	-0.516989
<b>15</b>	calls_outgoing_count	-0.513689
<b>17</b>	calls_outgoing_duration	-0.502408
<b>7</b>	user_spendings	-0.500438