In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import os
import sklearn
import matplotlib as ml
%matplotlib inline
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning: numpy.alen is deprecated, use len instead
  from numpy import alen, sum, array, zeros, ones, asarray, interp, random
```

Dataset 1 Parkinson's disease classification

In [2]:
```
df=pd.read_csv('/content/drive/My Drive/parkinsons.data')
```

In [3]:
```
df.shape
```

Out[3]:
```
(195, 24)
```

In [4]:
```
df
```

Out[4]:

	name	gender	TRA	TRA	DFAD	nonfiveo	nonfivehiver	nonfivehiffiver	nohiffivehiffiver	hifiveo	hi
0	b	1	0.00662	0.7500	0.57337	200	200	0.000000	0.000000	0.000000	0.000000
1	b	1	0.00100	0.7500	0.60929	200	200	0.000000	0.000000	0.000000	0.000000
2	b	1	0.00240	0.6000	0.54868	200	200	0.000000	0.000000	0.000000	0.000000
3	b	1	0.00290	0.8000	0.60267	199	197	0.010000	0.000000	0.000000	0.000000
4	b	1	0.00568	0.7000	0.62267	200	200	0.000000	0.000000	0.000000	0.000000
191	jh	0	0.00293	0.6000	0.60148	417	412	0.010407	0.000000	0.000000	0.000000
192	jh	0	0.00268	0.6667	0.62123	387	386	0.002584	0.000000	0.000000	0.000000
193	jh	0	0.00246	0.5000	0.60475	387	386	0.002584	0.000000	0.000000	0.000000
194	jh	0	0.00239	0.5000	0.59853	390	389	0.002570	0.000000	0.000000	0.000000
195	jh	0	0.00554	0.8667	0.64274	390	389	0.002570	0.000000	0.000000	0.000000

195 rows × 24 columns

In [5]:
```
df.to_csv('/content/drive/My Drive/', index=False)
```

Training data: 1 Validing data count: 0 Unique value

id	int	1	100
age	int	1	100
chbp	int	1	100
dm	int	1	100
chol	int	1	100
htn	int	1	100
cmbrthdsd	int	1	100
hxmvdsd	int	1	100
stenosis	int	1	100
diffdistal	int	1	100
diffprox	int	1	100
diffmed	int	1	100
diffdiag	int	1	100
diffstent	int	1	100
diabetes	int	1	100
smoker	int	1	100
priormi	int	1	100
hxmyo	int	1	100
hxpci	int	1	100
hxcabg	int	1	100
hxhtn	int	1	100
hxchf	int	1	100
hxdiabetes	int	1	100
hxsmoker	int	1	100
hxrenfail	int	1	100
hxpvd	int	1	100
hxpulmdisease	int	1	100
hxstrokecerebrovascdis	int	1	100
hxcardiomyopathy	int	1	100
hxmitralregurg	int	1	100
hxaortvalvedisease	int	1	100
hxmalignancy	int	1	100
hxpriorhtn	int	1	100
hxpriorchf	int	1	100
hxpriordiabetes	int	1	100
hxpriorsmoker	int	1	100
hxpriorrenfail	int	1	100
hxpriorstroke	int	1	100
hxpriorpvd	int	1	100
hxpriorpulmdisease	int	1	100
hxpriorstrokecerebrovascdis	int	1	100
hxpriorcardiomyopathy	int	1	100
hxpriormitralregurg	int	1	100
hxprioraortvalvedisease	int	1	100
hxpriormalignancy	int	1	100
hxpriormi	int	1	100
hxpriorpci	int	1	100
hxpriorcabg	int	1	100
hxpriorhxmyo	int	1	100
hxpriorhxmvdsd	int	1	100
hxpriorhxcmbrthdsd	int	1	100
hxpriorhxhtn	int	1	100
hxpriorhxchf	int	1	100
hxpriorhxdiabetes	int	1	100
hxpriorhxsmoker	int	1	100
hxpriorhxrenfail	int	1	100
hxpriorhxpvd	int	1	100
hxpriorhxpulmdisease	int	1	100
hxpriorhxstrokecerebrovascdis	int	1	100
hxpriorhxcardiomyopathy	int	1	100
hxpriorhxmitralregurg	int	1	100
hxpriorhxaortvalvedisease	int	1	100
hxpriorhxmalignancy	int	1	100
hxpriorhxmi	int	1	100
hxpriorhxpci	int	1	100
hxpriorhxcabg	int	1	100
hxpriorhxhxmyo	int	1	100
hxpriorhxhxmvdsd	int	1	100
hxpriorhxhxcmbrthdsd	int	1	100
hxpriorhxhxhtn	int	1	100
hxpriorhxhxchf	int	1	100
hxpriorhxhxdiabetes	int	1	100
hxpriorhxhxsmoker	int	1	100
hxpriorhxhxrenfail	int	1	100
hxpriorhxhxpvd	int	1	100
hxpriorhxhxpulmdisease	int	1	100
hxpriorhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxcardiomyopathy	int	1	100
hxpriorhxhxmitralregurg	int	1	100
hxpriorhxhxaortvalvedisease	int	1	100
hxpriorhxhxmalignancy	int	1	100
hxpriorhxhxmi	int	1	100
hxpriorhxhxpci	int	1	100
hxpriorhxhxcabg	int	1	100
hxpriorhxhxhxmyo	int	1	100
hxpriorhxhxhxmvdsd	int	1	100
hxpriorhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhtn	int	1	100
hxpriorhxhxhxchf	int	1	100
hxpriorhxhxhxdiabetes	int	1	100
hxpriorhxhxhxsmoker	int	1	100
hxpriorhxhxhxrenfail	int	1	100
hxpriorhxhxhxpvd	int	1	100
hxpriorhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxmalignancy	int	1	100
hxpriorhxhxhxmi	int	1	100
hxpriorhxhxhxpci	int	1	100
hxpriorhxhxhxcabg	int	1	100
hxpriorhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxchf	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxdiabetes	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxsmoker	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxrenfail	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpvd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpulmdisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxstrokecerebrovascdis	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcardiomyopathy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmitralregurg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxaortvalvedisease	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmalignancy	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmi	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxpci	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcabg	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmyo	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxmvdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxcmbrthdsd	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhxhtn	int	1	100
hxpriorhxhxhxhxhxhxhxhxhxhxhxhxhx

bpd_interfaces_ex_34	int	0	198
bpd_interfaces_ex_35	int	0	198
bpd_interfaces_ex_36	int	0	198
bpd_interfaces_ex_37	int	0	197
bpd_interfaces_ex_38	int	0	197
bpd_interfaces_ex_39	int	0	197
bpd_interfaces_ex_40	int	0	195
bpd_interfaces_ex_41	int	0	195
bpd_interfaces_ex_42	int	0	195
bpd_interfaces_ex_43	int	0	195

In [ ]: df.dtypes

```
Out[ ]: age                                   int32
        sex                                   int32
        cp                                    int32
        trtbps                                int32
        chol                                  int32
        fbs                                   int32
        restecg                               int32
        thalachh                              int32
        exng                                  int32
        oldpeak                               int32
        slp                                   int32
        caa                                   int32
        thall                                 int32
        output                                int32
        dtype: object
```

In [ ]: df.describe()

```
Out[ ]:                age         sex          cp       trtbps        chol         fbs     restecg    thalachh        exng     oldpeak         slp         caa       thall      output
        count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
        mean    54.366337    0.683168    0.966997  131.623762  246.264026    0.148515    0.528053  149.646865    0.326733    1.039604    1.399340    0.729373    2.313531    0.544554
        std      9.082101    0.466011    1.032052   17.538143   51.830751    0.356198    0.525860   22.905161    0.469794    1.161075    0.616226    1.022606    0.608075    0.498835
        min     29.000000    0.000000    0.000000   94.000000  126.000000    0.000000    0.000000   71.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
        25%     47.500000    0.000000    0.000000  120.000000  211.000000    0.000000    0.000000  133.500000    0.000000    0.000000    1.000000    0.000000    2.000000    0.000000
        50%     55.000000    1.000000    1.000000  130.000000  240.000000    0.000000    1.000000  153.000000    0.000000    0.800000    1.000000    0.000000    2.000000    1.000000
        75%     61.000000    1.000000    2.000000  140.000000  274.500000    0.000000    1.000000  166.000000    1.000000    1.600000    2.000000    1.000000    3.000000    1.000000
        max     77.000000    1.000000    3.000000  200.000000  564.000000    1.000000    2.000000  202.000000    1.000000    6.200000    2.000000    4.000000    3.000000    1.000000
```

In [ ]: df.shape

```
Out[ ]: (303, 14)
```

In [ ]: df.isnull()

```
Out[ ]:      age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  caa  thall  output
        0  False  NaN  NaN   NaN   NaN  NaN   NaN   NaN   NaN   NaN  NaN  NaN  NaN  NaN
        1  False  NaN  NaN   NaN   NaN  NaN   NaN   NaN   NaN   NaN  NaN  NaN  NaN  NaN
        2  False  NaN  NaN   NaN   NaN  NaN   NaN   NaN   NaN   NaN  NaN  NaN  NaN  NaN
```

Applying standard scaler on all columns to make sure they all are in the same range

In [ ]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
X = pd.DataFrame(X,columns=X.columns)
```

Using Chi-Informed Multiple choices the features while combining the class labels. The number of features produced by LDA number of classes minus one. Since there are two classes, we are left with one column. Combining the class labels with multiple features results in a single feature vector.

In [ ]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=1)
X = lda.fit(X, y).transform(X)
```

In [ ]:
```
X.shape
```

Out[ ]:
```
(1000, 1)
```

In [ ]:
```
X = pd.DataFrame(X,columns=['Feature_1'])
```

In [ ]:
```
X.head()
```

Out[ ]:

	Feature_1
0	0.241148
1	0.241148
2	0.241148
3	0.241148
4	0.241148

In [ ]:
```
y.head()
```

Out[ ]:
```
0    0
1    0
2    0
3    0
4    0
Name: Class, dtype: int64
```

In [ ]:
```
X.shape, y.shape
```

Out[ ]:
```
((1000, 1), (1000,))
```

In [ ]:
```
X.to_csv('Feature.csv')
y.to_csv('Class.csv')
```

In [ ]:
```
X = pd.read_csv('Feature.csv')
y = pd.read_csv('Class.csv')
```

In [ ]:
```
X.head()
```

Out[ ]:

	Unnamed: 0	Feature_1
0	0	0.241148
1	1	0.241148
2	2	0.241148
3	3	0.241148
4	4	0.241148

In [ ]:
```
y.head()
```

Out[ ]:

	Unnamed: 0	Class
0	0	0
1	1	0
2	2	0
3	3	0
4	4	0

Total rows: 2 columns

In [ ]:
```
%pip install -q -U keras-nlp tensorflow-text==2.15.0 keras Pillow,cv2,numpy,tqdm,'ipywidgets<8'
!apt-get -qq -y -f install
import os, sys
os.environ["KERAS_BACKEND"] = "jax"
os.environ["XLA_FLAGS"] = "--xla_gpu_cuda_data_dir=/usr/lib/cuda"
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.manifold.t_sne import TSNE
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.decomposition import LinearDiscriminantAnalysis
from sklearn.cross_validation import StratifiedKFold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.naive_bayes import CategoricalNB
from sklearn.naive_bayes import MultinomialNaiveBayes
from sklearn.naive_bayes import BernoulliNaiveBayes
from sklearn.naive_bayes import ComplementNaiveBayes
from sklearn.naive_bayes import CategoricalNaiveBayes
from sklearn.naive_bayes import GaussianNaiveBayes
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.linear_model import RidgeClassifierCV
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDOneClassSVM
from sklearn.linear_model import HuberRegressor
from sklearn.linear_model import LinearSVC
from sklearn.linear_model import NuSVC
from sklearn.linear_model import SVC
from sklearn.linear_model import KNeighborsClassifier
from sklearn.linear_model import RadiusNeighborsClassifier
from sklearn.linear_model import NearestCentroid
from sklearn.linear_model import KernelDensity
from sklearn.linear_model import LocalOutlierFactor
from sklearn.linear_model import IsolationForest
from sklearn.linear_model import MinCovDet
from sklearn.linear_model import EllipticEnvelope
from sklearn.linear_model import RandomForestClassifier
from sklearn.linear_model import ExtraTreesClassifier
from sklearn.linear_model import AdaBoostClassifier
from sklearn.linear_model import GradientBoostingClassifier
from sklearn.linear_model import HistGradientBoostingClassifier
from sklearn.linear_model import BaggingClassifier
from sklearn.linear_model import VotingClassifier
from sklearn.linear_model import StackingClassifier
from sklearn.linear_model import RandomTreesEmbedding
from sklearn.linear_model import GaussianProcessClassifier
from sklearn.linear_model import LabelPropagation
from sklearn.linear_model import LabelSpreading
from sklearn.linear_model import DecisionTreeClassifier
from sklearn.linear_model import ExtraTreeClassifier
from sklearn.linear_model import DummyClassifier
from sklearn.linear_model import MLPClassifier
from sklearn.linear_model import GaussianNB
from sklearn.linear_model import MultinomialNB
from sklearn.linear_model import BernoulliNB
from sklearn.linear_model import ComplementNB
from sklearn.linear_model import CategoricalNB
from sklearn.linear_model import MultinomialNaiveBayes
from sklearn.linear_model import BernoulliNaiveBayes
from sklearn.linear_model import ComplementNaiveBayes
from sklearn.linear_model import CategoricalNaiveBayes
from sklearn.linear_model import GaussianNaiveBayes
from sklearn.linear_model import LinearDiscriminantAnalysis
from sklearn.linear_model import QuadraticDiscriminantAnalysis
from sklearn.linear_model import KernelPCA
from sklearn.linear_model import IncrementalPCA
from sklearn.linear_model import SparsePCA
from sklearn.linear_model import TruncatedSVD
from sklearn.linear_model import FastICA
from sklearn.linear_model import NMF
from sklearn.linear_model import LatentDirichletAllocation
from sklearn.linear_model import MiniBatchNMF
from sklearn.linear_model import FactorAnalysis
from sklearn.linear_model import PCA
from sklearn.linear_model import SelectKBest
from sklearn.linear_model import chi2
from sklearn.linear_model import f_classif
from sklearn.linear_model import mutual_info_classif
from sklearn.linear_model import f_regression
from sklearn.linear_model import mutual_info_regression
from sklearn.linear_model import SelectPercentile
from sklearn.linear_model import GenericUnivariateSelect
from sklearn.linear_model import VarianceThreshold
from sklearn.linear_model import SelectFromModel
from sklearn.linear_model import SelectFromModelCV
from sklearn.linear_model import RFE
from sklearn.linear_model import RFECV
from sklearn.linear_model import SequentialFeatureSelector
from sklearn.linear_model import GridSearchCV
from sklearn.linear_model import HalvingGridSearchCV
from sklearn.linear_model import RandomizedSearchCV
from sklearn.linear_model import HalvingRandomSearchCV
from sklearn.linear_model import ParameterGrid
from sklearn.linear_model import ParameterSampler
from sklearn.linear_model import PredefinedSplit
from sklearn.linear_model import TunedThresholdClassifierCV
from sklearn.linear_model import LabelEncoder
from sklearn.linear_model import LabelBinarizer
from sklearn.linear_model import MultiLabelBinarizer
from sklearn.linear_model import OneHotEncoder
from sklearn.linear_model import OrdinalEncoder
from sklearn.linear_model import TargetEncoder
from sklearn.linear_model import KBinsDiscretizer
from sklearn.linear_model import FeatureHasher
from sklearn.linear_model import DictVectorizer
from sklearn.linear_model import PolynomialFeatures
from sklearn.linear_model import SplineTransformer
from sklearn.linear_model import Normalizer
from sklearn.linear_model import QuantileTransformer
from sklearn.linear_model import PowerTransformer
from sklearn.linear_model import MaxAbsScaler
from sklearn.linear_model import MinMaxScaler
from sklearn.linear_model import StandardScaler
from sklearn.linear_model import RobustScaler
from sklearn.linear_model import CountVectorizer
from sklearn.linear_model import HashingVectorizer
from sklearn.linear_model import TfidfTransformer
from sklearn.linear_model import TfidfVectorizer
from sklearn.linear_model import Tokenizer
from sklearn.linear_model import pad_sequences
from sklearn.linear_model import to_categorical
from sklearn.linear_model import plot_model
from sklearn.linear_model import Sequential
from sklearn.linear_model import Input
from sklearn.linear_model import Dense
from sklearn.linear_model import Dropout
from sklearn.linear_model import Activation
from sklearn.linear_model import Embedding
from sklearn.linear_model import LSTM
from sklearn.linear_model import Bidirectional
from sklearn.linear_model import GlobalMaxPooling1D
from sklearn.linear_model import GlobalAveragePooling1D
from sklearn.linear_model import MaxPooling1D
from sklearn.linear_model import AveragePooling1D
from sklearn.linear_model import Conv1D
from sklearn.linear_model import SeparableConv1D
from sklearn.linear_model import DepthwiseConv1D
from sklearn.linear_model import GlobalMaxPooling2D
from sklearn.linear_model import GlobalAveragePooling2D
from sklearn.linear_model import MaxPooling2D
from sklearn.linear_model import AveragePooling2D
from sklearn.linear_model import Conv2D
from sklearn.linear_model import SeparableConv2D
from sklearn.linear_model import DepthwiseConv2D
from sklearn.linear_model import GlobalMaxPooling3D
from sklearn.linear_model import GlobalAveragePooling3D
from sklearn.linear_model import MaxPooling3D
from sklearn.linear_model import AveragePooling3D
from sklearn.linear_model import Conv3D
from sklearn.linear_model import SeparableConv3D
from sklearn.linear_model import DepthwiseConv3D
from sklearn.linear_model import Flatten
from sklearn.linear_model import Reshape
from sklearn.linear_model import Permute
from sklearn.linear_model import RepeatVector
from sklearn.linear_model import Lambda
from sklearn.linear_model import Masking
from sklearn.linear_model import Cropping1D
from sklearn.linear_model import Cropping2D
from sklearn.linear_model import Cropping3D
from sklearn.linear_model import UpSampling1D
from sklearn.linear_model import UpSampling2D
from sklearn.linear_model import UpSampling3D
from sklearn.linear_model import ZeroPadding1D
from sklearn.linear_model import ZeroPadding2D
from sklearn.linear_model import ZeroPadding3D
from sklearn.linear_model import Sequential
from sklearn.linear_model import Input
from sklearn.linear_model import Dense
from sklearn.linear_model import Dropout
from sklearn.linear_model import Activation
from sklearn.linear_model import Embedding
from sklearn.linear_model import LSTM
from sklearn.linear_model import Bidirectional
from sklearn.linear_model import GlobalMaxPooling1D
from sklearn.linear_model import GlobalAveragePooling1D
from sklearn.linear_model import MaxPooling1D
from sklearn.linear_model import AveragePooling1D
from sklearn.linear_model import Conv1D
from sklearn.linear_model import SeparableConv1D
from sklearn.linear_model import DepthwiseConv1D
from sklearn.linear_model import GlobalMaxPooling2D
from sklearn.linear_model import GlobalAveragePooling2D
from sklearn.linear_model import MaxPooling2D
from sklearn.linear_model import AveragePooling2D
from sklearn.linear_model import Conv2D
from sklearn.linear_model import SeparableConv2D
from sklearn.linear_model import DepthwiseConv2D
from sklearn.linear_model import GlobalMaxPooling3D
from sklearn.linear_model import GlobalAveragePooling3D
from sklearn.linear_model import MaxPooling3D
from sklearn.linear_model import AveragePooling3D
from sklearn.linear_model import Conv3D
from sklearn.linear_model import SeparableConv3D
from sklearn.linear_model import DepthwiseConv3D
from sklearn.linear_model import Flatten
from sklearn.linear_model import Reshape
from sklearn.linear_model import Permute
from sklearn.linear_model import RepeatVector
from sklearn.linear_model import Lambda
from sklearn.linear_model import Masking
from sklearn.linear_model import Cropping1D
from sklearn.linear_model import Cropping2D
from sklearn.linear_model import Cropping3D
from sklearn.linear_model import UpSampling1D
from sklearn.linear_model import UpSampling2D
from sklearn.linear_model import UpSampling3D
from sklearn.linear_model import ZeroPadding1D
from sklearn.linear_model import ZeroPadding2D
from sklearn.linear_model import ZeroPadding3D
from sklearn.linear_model import Adam
from sklearn.linear_model import Adamax
from sklearn.linear_model import Adadelta
from sklearn.linear_model import Adagrad
from sklearn.linear_model import AdamW
from sklearn.linear_model import Adafactor
from sklearn.linear_model import Nadam
from sklearn.linear_model import RMSprop
from sklearn.linear_model import SGD
from sklearn.linear_model import EarlyStopping
from sklearn.linear_model import ReduceLROnPlateau
from sklearn.linear_model import LearningRateScheduler
from sklearn.linear_model import CSVLogger
from sklearn.linear_model import ProgbarLogger
from sklearn.linear_model import TerminateOnNaN
from sklearn.linear_model import RemoteMonitor
from sklearn.linear_model import LambdaCallback
from sklearn.linear_model import TensorBoard
from sklearn.linear_model import ModelCheckpoint
from sklearn.linear_model import BackupAndRestore
from sklearn.linear_model import EarlyStoppingAtMinLoss
from sklearn.linear_model import StopAtNaN
from sklearn.linear_model import LossNotImproving
from sklearn.linear_model import RelativeDifference
from sklearn.linear_model import NumberOfWeightsChanged
from sklearn.linear_model import UpdateCheckpoint
from sklearn.linear_model import ClassificationOutputs
from sklearn.linear_model import SaveBestFilepath
from sklearn.linear_model import SaveAllFilepaths
from sklearn.linear_model import ReduceLROnPlateauOnPlateau
from sklearn.linear_model import ReduceLROnPlateauOnPlateauWithWarmup
from sklearn.linear_model import ReduceLROnPlateauOnPlateauWithWarmupAndMin
from sklearn.linear_model import ReduceLROnPlateauOnPlateauWithWarmupAndMinAndMax
```

In [ ]:
```
!cp /kaggle/input/*/*/*/Feature.csv /kaggle/input/*/*/*/Class.csv /kaggle/working/
```

In [ ]:
```
X = pd.read_csv('Feature.csv')
y = pd.read_csv('Class.csv')
```

In [ ]:
```
X.shape, y.shape
```

Out[ ]:
```
((1000, 2), (1000, 2))
```

In [ ]:
```
X.head()
```

Out[ ]:

	Unnamed: 0	Feature_1
0	0	0.241148
1	1	0.241148
2	2	0.241148
3	3	0.241148
4	4	0.241148

In [ ]:
```
y.head()
```

Out[ ]:

	Unnamed: 0	Class
0	0	0
1	1	0
2	2	0
3	3	0
4	4	0

Total rows: 2 columns

In [ ]:
```
%pip install -q -U keras-nlp tensorflow-text==2.15.0 keras Pillow,cv2,numpy,tqdm,'ipywidgets<8'
!apt-get -qq -y -f install
import os, sys
os.environ["KERAS_BACKEND"] = "jax"
os.environ["XLA_FLAGS"] = "--xla_gpu_cuda_data_dir=/usr/lib/cuda"
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.manifold.t_sne import TSNE
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.decomposition import LinearDiscriminantAnalysis
from sklearn.cross_validation import StratifiedKFold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.naive_bayes import CategoricalNB
from sklearn.naive_bayes import MultinomialNaiveBayes
from sklearn.naive_bayes import BernoulliNaiveBayes
from sklearn.naive_bayes import ComplementNaiveBayes
from sklearn.naive_bayes import CategoricalNaiveBayes
from sklearn.naive_bayes import GaussianNaiveBayes
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.linear_model import RidgeClassifierCV
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDOneClassSVM
from sklearn.linear_model import HuberRegressor
from sklearn.linear_model import LinearSVC
from sklearn.linear_model import NuSVC
from sklearn.linear_model import SVC
from sklearn.linear_model import KNeighborsClassifier
from sklearn.linear_model import RadiusNeighborsClassifier
from sklearn.linear_model import NearestCentroid
from sklearn.linear_model import KernelDensity
from sklearn.linear_model import LocalOutlierFactor
from sklearn.linear_model import IsolationForest
from sklearn.linear_model import MinCovDet
from sklearn.linear_model import EllipticEnvelope
from sklearn.linear_model import RandomForestClassifier
from sklearn.linear_model import ExtraTreesClassifier
from sklearn.linear_model import AdaBoostClassifier
from sklearn.linear_model import GradientBoostingClassifier
from sklearn.linear_model import HistGradientBoostingClassifier
from sklearn.linear_model import BaggingClassifier
from sklearn.linear_model import VotingClassifier
from sklearn.linear_model import StackingClassifier
from sklearn.linear_model import RandomTreesEmbedding
from sklearn.linear_model import GaussianProcessClassifier
from sklearn.linear_model import LabelPropagation
from sklearn.linear_model import LabelSpreading
from sklearn.linear_model import DecisionTreeClassifier
from sklearn.linear_model import ExtraTreeClassifier
from sklearn.linear_model import DummyClassifier
from sklearn.linear_model import MLPClassifier
from sklearn.linear_model import GaussianNB
from sklearn.linear_model import MultinomialNB
from sklearn.linear_model import BernoulliNB
from sklearn.linear_model import ComplementNB
from sklearn.linear_model import CategoricalNB
from sklearn.linear_model import MultinomialNaiveBayes
from sklearn.linear_model import BernoulliNaiveBayes
from sklearn.linear_model import ComplementNaiveBayes
from sklearn.linear_model import CategoricalNaiveBayes
from sklearn.linear_model import GaussianNaiveBayes
from sklearn.linear_model import LinearDiscriminantAnalysis
from sklearn.linear_model import QuadraticDiscriminantAnalysis
from sklearn.linear_model import KernelPCA
from sklearn.linear_model import IncrementalPCA
from sklearn.linear_model import SparsePCA
from sklearn.linear_model import TruncatedSVD
from sklearn.linear_model import FastICA
from sklearn.linear_model import NMF
from sklearn.linear_model import LatentDirichletAllocation
from sklearn.linear_model import MiniBatchNMF
from sklearn.linear_model import FactorAnalysis
from sklearn.linear_model import PCA
from sklearn.linear_model import SelectKBest
from sklearn.linear_model import chi2
from sklearn.linear_model import f_classif
from sklearn.linear_model import mutual_info_classif
from sklearn.linear_model import f_regression
from sklearn.linear_model import mutual_info_regression
from sklearn.linear_model import SelectPercentile
from sklearn.linear_model import GenericUnivariateSelect
from sklearn.linear_model import VarianceThreshold
from sklearn.linear_model import SelectFromModel
from sklearn.linear_model import SelectFromModelCV
from sklearn.linear_model import RFE
from sklearn.linear_model import RFECV
from sklearn.linear_model import SequentialFeatureSelector
from sklearn.linear_model import GridSearchCV
from sklearn.linear_model import HalvingGridSearchCV
from sklearn.linear_model import RandomizedSearchCV
from sklearn.linear_model import HalvingRandomSearchCV
from sklearn.linear_model import ParameterGrid
from sklearn.linear_model import ParameterSampler
from sklearn.linear_model import PredefinedSplit
from sklearn.linear_model import TunedThresholdClassifierCV
from sklearn.linear_model import LabelEncoder
from sklearn.linear_model import LabelBinarizer
from sklearn.linear_model import MultiLabelBinarizer
from sklearn.linear_model import OneHotEncoder
from sklearn.linear_model import OrdinalEncoder
from sklearn.linear_model import TargetEncoder
from sklearn.linear_model import KBinsDiscretizer
from sklearn.linear_model import FeatureHasher
from sklearn.linear_model import DictVectorizer
from sklearn.linear_model import PolynomialFeatures
from sklearn.linear_model import SplineTransformer
from sklearn.linear_model import Normalizer
from sklearn.linear_model import QuantileTransformer
from sklearn.linear_model import PowerTransformer
from sklearn.linear_model import MaxAbsScaler
from sklearn.linear_model import MinMaxScaler
from sklearn.linear_model import StandardScaler
from sklearn.linear_model import RobustScaler
from sklearn.linear_model import CountVectorizer
from sklearn.linear_model import HashingVectorizer
from sklearn.linear_model import TfidfTransformer
from sklearn.linear_model import TfidfVectorizer
from sklearn.linear_model import Tokenizer
from sklearn.linear_model import pad_sequences
from sklearn.linear_model import to_categorical
from sklearn.linear_model import plot_model
from sklearn.linear_model import Sequential
from sklearn.linear_model import Input
from sklearn.linear_model import Dense
from sklearn.linear_model import Dropout
from sklearn.linear_model import Activation
from sklearn.linear_model import Embedding
from sklearn.linear_model import LSTM
from sklearn.linear_model import Bidirectional
from sklearn.linear_model import GlobalMaxPooling1D
from sklearn.linear_model import GlobalAveragePooling1D
from sklearn.linear_model import MaxPooling1D
from sklearn.linear_model import AveragePooling1D
from sklearn.linear_model import Conv1D
from sklearn.linear_model import SeparableConv1D
from sklearn.linear_model import DepthwiseConv1D
from sklearn.linear_model import GlobalMaxPooling2D
from sklearn.linear_model import GlobalAveragePooling2D
from sklearn.linear_model import MaxPooling2D
from sklearn.linear_model import AveragePooling2D
from sklearn.linear_model import Conv2D
from sklearn.linear_model import SeparableConv2D
from sklearn.linear_model import DepthwiseConv2D
from sklearn.linear_model import GlobalMaxPooling3D
from sklearn.linear_model import GlobalAveragePooling3D
from sklearn.linear_model import MaxPooling3D
from sklearn.linear_model import AveragePooling3D
from sklearn.linear_model import Conv3D
from sklearn.linear_model import SeparableConv3D
from sklearn.linear_model import DepthwiseConv3D
from sklearn.linear_model import Flatten
from sklearn.linear_model import Reshape
from sklearn.linear_model import Permute
from sklearn.linear_model import RepeatVector
from sklearn.linear_model import Lambda
from sklearn.linear_model import Masking
from sklearn.linear_model import Cropping1D
from sklearn.linear_model import Cropping2D
from sklearn.linear_model import Cropping3D
from sklearn.linear_model import UpSampling1D
from sklearn.linear_model import UpSampling2D
from sklearn.linear_model import UpSampling3D
from sklearn.linear_model import ZeroPadding1D
from sklearn.linear_model import ZeroPadding2D
from sklearn.linear_model import ZeroPadding3D
from sklearn.linear_model import Sequential
from sklearn.linear_model import Input
from sklearn.linear_model import Dense
from sklearn.linear_model import Dropout
from sklearn.linear_model import Activation
from sklearn.linear_model import Embedding
from sklearn.linear_model import LSTM
from sklearn.linear_model import Bidirectional
from sklearn.linear_model import GlobalMaxPooling1D
from sklearn.linear_model import GlobalAveragePooling1D
from sklearn.linear_model import MaxPooling1D
from sklearn.linear_model import AveragePooling1D
from sklearn.linear_model import Conv1D
from sklearn.linear_model import SeparableConv1D
from sklearn.linear_model import DepthwiseConv1D
from sklearn.linear_model import GlobalMaxPooling2D
from sklearn.linear_model import GlobalAveragePooling2D
from sklearn.linear_model import MaxPooling2D
from sklearn.linear_model import AveragePooling2D
from sklearn.linear_model import Conv2D
from sklearn.linear_model import SeparableConv2D
from sklearn.linear_model import DepthwiseConv2D
from sklearn.linear_model import GlobalMaxPooling3D
from sklearn.linear_model import GlobalAveragePooling3D
from sklearn.linear_model import MaxPooling3D
from sklearn.linear_model import AveragePooling3D
from sklearn.linear_model import Conv3D
from sklearn.linear_model import SeparableConv3D
from sklearn.linear_model import DepthwiseConv3D
from sklearn.linear_model import Flatten
from sklearn.linear_model import Reshape
from sklearn.linear_model import Permute
from sklearn.linear_model import RepeatVector
from sklearn.linear_model import Lambda
from sklearn.linear_model import Masking
from sklearn.linear_model import Cropping1D
from sklearn.linear_model import Cropping2D
from sklearn.linear_model import Cropping3D
from sklearn.linear_model import UpSampling1D
from sklearn.linear_model import UpSampling2D
from sklearn.linear_model import UpSampling3D
from sklearn.linear_model import ZeroPadding1D
from sklearn.linear_model import ZeroPadding2D
from sklearn.linear_model import ZeroPadding3D
from sklearn.linear_model import Adam
from sklearn.linear_model import Adamax
from sklearn.linear_model import Adadelta
from sklearn.linear_model import Adagrad
from sklearn.linear_model import AdamW
from sklearn.linear_model import Adafactor
from sklearn.linear_model import Nadam
from sklearn.linear_model import RMSprop
from sklearn.linear_model import SGD
from sklearn.linear_model import EarlyStopping
from sklearn.linear_model import ReduceLROnPlateau
from sklearn.linear_model import LearningRateScheduler
from sklearn.linear_model import CSVLogger
from sklearn.linear_model import ProgbarLogger
from sklearn.linear_model import TerminateOnNaN
from sklearn.linear_model import RemoteMonitor
from sklearn.linear_model import LambdaCallback
from sklearn.linear_model import TensorBoard
from sklearn.linear_model import ModelCheckpoint
from sklearn.linear_model import BackupAndRestore
from sklearn.linear_model import EarlyStoppingAtMinLoss
from sklearn.linear_model import StopAtNaN
from sklearn.linear_model import LossNotImproving
from sklearn.linear_model import RelativeDifference
from sklearn.linear_model import NumberOfWeightsChanged
from sklearn.linear_model import UpdateCheckpoint
from sklearn.linear_model import ClassificationOutputs
from sklearn.linear_model import SaveBestFilepath
from sklearn.linear_model import SaveAllFilepaths
from sklearn.linear_model import ReduceLROnPlateauOnPlateau
from sklearn.linear_model import ReduceLROnPlateauOnPlateauWithWarmup
from sklearn.linear_model import ReduceLROnPlateauOnPlateauWithWarmupAndMin
from sklearn.linear_model import ReduceLROnPlateauOnPlateauWithWarmupAndMinAndMax
```

In [ ]:
```
!cp /kaggle/input/*/*/*/Feature.csv /kaggle/input/*/*/*/Class.csv /kaggle/working/
```

In [ ]:
```
X = pd.read_csv('Feature.csv')
y = pd.read_csv('Class.csv')
```

In [ ]:
```
X.shape, y.shape
```

Out[ ]:
```
((1000, 2), (1000, 2))
```

In [ ]:
```
X.head()
```

Out[ ]:

	Unnamed: 0	Feature_1
0	0	0.241148
1	1	0.241148
2	2	0.241148
3	3	0.241148
4	4	0.241148

In [ ]:
```
y.head()
```

Out[ ]:

	Unnamed: 0	Class
0	0	0
1	1	0
2	2	0
3	3	0
4	4	0

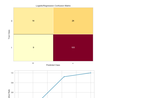
Total rows: 2 columns

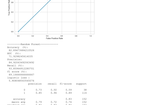
In [ ]:
```
%pip install -q -U keras-nlp tensorflow-text==2.15.0 keras Pillow,cv2,numpy,tqdm,'ipywidgets<8'
!apt-get -qq -y -f install
import os, sys
os.environ["KERAS_BACKEND"] = "jax"
os.environ["XLA_FLAGS"] = "--xla_gpu_cuda_data_dir=/usr/lib/cuda"
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.manifold.t_sne import TSNE
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.decomposition import LinearDiscriminantAnalysis
from sklearn.cross_validation import StratifiedKFold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.naive_bayes import CategoricalNB
from sklearn.naive_bayes import MultinomialNaiveBayes
from sklearn.naive_bayes import BernoulliNaiveBayes
from sklearn.naive_bayes import ComplementNaiveBayes
from sklearn.naive_bayes import CategoricalNaiveBayes
from sklearn.naive_bayes import GaussianNaiveBayes
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.linear_model import RidgeClassifierCV
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDOneClassSVM
from sklearn.linear_model import HuberRegressor
from sklearn.linear_model import LinearSVC
from sklearn.linear_model import NuSVC
from sklearn.linear_model import SVC
from sklearn.linear_model import KNeighborsClassifier
from sklearn.linear_model import RadiusNeighborsClassifier
from sklearn.linear_model import NearestCentroid
from sklearn.linear_model import KernelDensity
from sklearn.linear_model import LocalOutlierFactor
from sklearn.linear_model import IsolationForest
from sklearn.linear_model import MinCovDet
from sklearn.linear_model import EllipticEnvelope
from sklearn.linear_model import RandomForestClassifier
from sklearn.linear_model import ExtraTreesClassifier
from sklearn.linear_model import AdaBoostClassifier
from sklearn.linear_model import GradientBoostingClassifier
from sklearn.linear_model import HistGradientBoostingClassifier
from sklearn.linear_model import BaggingClassifier
from sklearn.linear_model import VotingClassifier
from sklearn.linear_model import StackingClassifier
from sklearn.linear_model import RandomTreesEmbedding
from sklearn.linear_model import GaussianProcessClassifier
from sklearn.linear_model import LabelPropagation
from sklearn.linear_model import LabelSpreading
from sklearn.linear_model import DecisionTreeClassifier
from sklearn.linear_model import ExtraTreeClassifier
from sklearn.linear_model import DummyClassifier
from sklearn.linear_model import MLPClassifier
from sklearn.linear_model import GaussianNB
from sklearn.linear_model import MultinomialNB
from sklearn.linear_model import BernoulliNB
from sklearn.linear_model import ComplementNB
from sklearn.linear_model import CategoricalNB
from sklearn.linear_model import MultinomialNaiveBayes
from sklearn.linear_model import BernoulliNaiveBayes
from sklearn.linear_model import ComplementNaiveBayes
from sklearn.linear_model import CategoricalNaiveBayes
from sklearn.linear_model import GaussianNaiveBayes
from sklearn.linear_model import LinearDiscriminantAnalysis
from sklearn.linear_model import QuadraticDiscriminantAnalysis
from sklearn.linear_model import KernelPCA
from sklearn.linear_model import IncrementalPCA
from sklearn.linear_model import SparsePCA
from sklearn.linear_model import TruncatedSVD
from sklearn.linear_model import FastICA
from sklearn.linear_model import NMF
from sklearn.linear_model import LatentDirichletAllocation
from sklearn.linear_model import MiniBatchNMF
from sklearn.linear_model import FactorAnalysis
from sklearn.linear_model import PCA
from sklearn.linear_model import SelectKBest
from sklearn.linear_model import chi2
from sklearn.linear_model import f_classif
from sklearn.linear_model import mutual_info_classif
from sklearn.linear_model import f_regression
from sklearn.linear_model import mutual_info_regression
from sklearn.linear_model import SelectPercentile
from sklearn.linear_model import GenericUnivariateSelect
from sklearn.linear_model import VarianceThreshold
from sklearn.linear_model import SelectFromModel
from sklearn.linear_model import SelectFromModelCV
from sklearn.linear_model import RFE
from sklearn.linear_model import RFECV
from sklearn.linear_model import SequentialFeatureSelector
from sklearn.linear_model import GridSearchCV
from sklearn.linear_model import HalvingGridSearchCV
from sklearn.linear_model import RandomizedSearchCV
from sklearn.linear_model import HalvingRandomSearchCV
from sklearn.linear_model import ParameterGrid
from sklearn.linear_model import ParameterSampler
from sklearn.linear_model import PredefinedSplit
from sklearn.linear_model import TunedThresholdClassifierCV
from sklearn.linear_model import LabelEncoder
from sklearn.linear_model import LabelBinarizer
from sklearn.linear_model import MultiLabelBinarizer
from sklearn.linear_model import OneHotEncoder
from sklearn.linear_model import OrdinalEncoder
from sklearn.linear_model import TargetEncoder
from sklearn.linear_model import KBinsDiscretizer
from sklearn.linear_model import FeatureHasher
from sklearn.linear_model import DictVectorizer
from sklearn.linear_model import PolynomialFeatures
from sklearn.linear_model import SplineTransformer
from sklearn.linear_model import Normalizer
from sklearn.linear_model import QuantileTransformer
from sklearn.linear_model import PowerTransformer
from sklearn.linear_model import MaxAbsScaler
from sklearn.linear_model import MinMaxScaler
from sklearn.linear_model import StandardScaler
from sklearn.linear_model import RobustScaler
from sklearn.linear_model import CountVectorizer
from sklearn.linear_model import HashingVectorizer
from sklearn.linear_model import TfidfTransformer
from sklearn.linear_model import TfidfVectorizer
from sklearn.linear_model import Tokenizer
from sklearn.linear_model import pad_sequences
from sklearn.linear_model import to_categorical
from sklearn.linear_model import plot_model
from sklearn.linear_model import Sequential
from sklearn.linear_model import Input
from sklearn.linear_model import Dense
from sklearn.linear_model import Dropout
from sklearn.linear_model import Activation
from sklearn.linear_model import Embedding
from sklearn.linear_model import LSTM
from sklearn.linear_model import Bidirectional
from sklearn.linear_model import GlobalMaxPooling1D
from sklearn.linear_model import GlobalAveragePooling1D
from sklearn.linear_model import MaxPooling1D
from sklearn.linear_model import AveragePooling1D
from sklearn.linear_model import Conv1D
from sklearn.linear_model import SeparableConv1D
from sklearn.linear_model import DepthwiseConv1D
from sklearn.linear_model import GlobalMaxPooling2D
from sklearn.linear_model import GlobalAveragePooling2D
from sklearn.linear_model import MaxPooling2D
from sklearn.linear_model import AveragePooling2D
from sklearn.linear_model import Conv2D
from sklearn.linear_model import SeparableConv2D
from sklearn.linear_model import DepthwiseConv2D
from sklearn.linear_model import GlobalMaxPooling3D
from sklearn.linear_model import GlobalAveragePooling3D
from sklearn.linear_model import MaxPooling3D
from sklearn.linear_model import AveragePooling3D
from sklearn.linear_model import Conv3D
from sklearn.linear_model import SeparableConv3D
from sklearn.linear_model import DepthwiseConv3D
from sklearn.linear_model import Flatten
from sklearn.linear_model import Reshape
from sklearn.linear_model import Permute
from sklearn.linear_model import RepeatVector
from sklearn.linear_model import Lambda
from sklearn.linear_model import Masking
from sklearn.linear_model import Cropping1D
from sklearn.linear_model import Cropping2D
from sklearn.linear_model import Cropping3D
from sklearn.linear_model import UpSampling1D
from sklearn.linear_model import UpSampling2D
from sklearn.linear_model import UpSampling3D
from sklearn.linear_model import ZeroPadding1D
from sklearn.linear_model import ZeroPadding2D
from sklearn.linear_model import ZeroPadding3D
from sklearn.linear_model import Sequential
from sklearn.linear_model import Input
from sklearn.linear_model import Dense
from sklearn.linear_model import Dropout
from sklearn.linear_model import Activation
from sklearn.linear_model import Embedding
from sklearn.linear_model import LSTM
from sklearn.linear_model import Bidirectional
from sklearn.linear_model import GlobalMaxPooling1D
from sklearn.linear_model import GlobalAveragePooling1D
from sklearn.linear_model import MaxPooling1D
from sklearn.linear_model import AveragePooling1D
from sklearn.linear_model import Conv1D
from sklearn.linear_model import SeparableConv1D
from sklearn.linear_model import DepthwiseConv1D
from sklearn.linear_model import GlobalMaxPooling2D
from sklearn.linear_model import GlobalAveragePooling2D
from sklearn.linear_model import MaxPooling2D
from sklearn.linear_model import AveragePooling2D
from sklearn.linear_model import Conv2D
from sklearn.linear_model import SeparableConv2D
from sklearn.linear_model import Dep
```

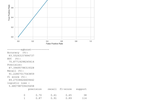
```
In[ ]: from dtlearn.preprocessing import Normalizer
       from dtlearn.ensemble.bagging import BaggingClassifier
       X, y = load_data(n=2000, d=2, random_state=42)
       X = X[:, np.newaxis, :]
       X.shape

       [Out]: (2000, 1, 2)
       [In]: np.unique(y)

       [Out]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
       [In]: from dtlearn.preprocessing import StandardScaler
             from dtlearn.ensemble.bagging import BaggingClassifier
             from sklearn.model_selection import train_test_split
             X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
             scaler = StandardScaler()
             X_train = scaler.fit_transform(X_train)
             X_test = scaler.transform(X_test)
             np.unique(y_train)

             [Out]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
             [In]: np.unique(y_test)

             [Out]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
             [In]: bagcls = BaggingClassifier(
                   base_estimator=None,
                   n_estimators=10,
                   max_samples=1.0,
                   max_features=1.0,
                   bootstrap=True,
                   bootstrap_features=False,
                   oob_score=False,
                   warm_start=False,
                   random_state=None,
                   verbose=0)
             [In]: bagcls.fit(X_train, y_train)

             [Out]: BaggingClassifier(base_estimator=None,
                    bootstrap=True,
                    bootstrap_features=False,
                    max_features=1.0,
                    max_samples=1.0,
                    n_estimators=10,
                    oob_score=False,
                    random_state=None,
                    verbose=0,
                    warm_start=False)
             [In]: bagcls.score(X_train, y_train)

             [Out]: 0.9975000000000001
             [In]: bagcls.score(X_test, y_test)

             [Out]: 0.9925
             [In]: bagcls.estimators_

             [Out]: [DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1968195386,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1650419051,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1814750379,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1548351926,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1962115311,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=2044897763,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1797859762,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1593904233,
                     splitter='best'),
                     DecisionTreeClassifier(class_weight=None,
                     criterion='gini',
                     max_depth=None,
                     max_features=None,
                     max_leaf_nodes=None,
                     min_impurity_decrease=0.0,
                     min_impurity_split=None,
                     min_samples_leaf=1,
                     min_samples_split=2,
                     min_weight_fraction_leaf=0.0,
                     presort=False,
                     random_state=1408636793,
                     splitter='best')]
             [In]: bagcls.n_features_

             [Out]: 2
             [In]: bagcls.n_classes_

             [Out]: 10
             [In]: bagcls.classes_

             [Out]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
             [In]: bagcls.base_estimator_

             [Out]: DecisionTreeClassifier(class_weight=None,
                    criterion='gini',
                    max_depth=None,
                    max_features=None,
                    max_leaf_nodes=None,
                    min_impurity_decrease=0.0,
                    min_impurity_split=None,
                    min_samples_leaf=1,
                    min_samples_split=2,
                    min_weight_fraction_leaf=0.0,
                    presort=False,
                    random_state=None,
                    splitter='best')
             [In]: bagcls.n_estimators

             [Out]: 10
             [In]: bagcls.max_samples

             [Out]: 1.0
             [In]: bagcls.max_features

             [Out]: 1.0
             [In]: bagcls.bootstrap

             [Out]: True
             [In]: bagcls.bootstrap_features

             [Out]: False
             [In]: bagcls.oob_score

             [Out]: False
             [In]: bagcls.warm_start

             [Out]: False
             [In]: bagcls.random_state

             [Out]: None
             [In]: bagcls.verbose

             [Out]: 0
             [In]: bagcls.estimator_params

             [Out]: ('base_estimator',
                     'bootstrap',
                     'bootstrap_features',
                     'max_features',
                     'max_samples',
                     'n_estimators',
                     'random_state',
                     'verbose')
             [In]: bagcls.oob_score_

             [Out]: 0.9975000000000001
             [In]: bagcls.oob_decision_function_

             [Out]: array([[1. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
                           [
```

```
In [ ]: fit_model_log_reg = LogisticRegressionCV(cv=5,max_iter=500,dual=False,n_jobs=-1,fit_intercept=True,
        )
        fit_model_log_reg.fit(X_train,y_train)
        print("Accuracy:",fit_model_log_reg.score(X_test,y_test))
        print("Cross Validation Score:",cross_val_score(fit_model_log_reg,X_train,y_train,cv=5,n_jobs=-1,
        )
        print("Cross Validation Mean Score:",cross_val_score(fit_model_log_reg,X_train,y_train,cv=5,n_jobs=
        )
        print("Classification report:\n",classification_report(y_test,fit_model_log_reg.predict(X_test)))
        print("Confusion matrix:\n",confusion_matrix(y_test,fit_model_log_reg.predict(X_test)))
        plot_confusion_matrix(fit_model_log_reg,X_test,y_test,cmap=plt.cmap_r,normalize=None)
        plot_confusion_matrix(fit_model_log_reg,X_test,y_test,cmap=plt.cmap_r,normalize='true')
        plot_roc_curve(fit_model_log_reg,X_test,y_test,)
        plot_precision_recall_curve(fit_model_log_reg,X_test,y_test,)
```

```
Accuracy: 0.9824561403508772
Cross Validation Score: [0.97802198 0.97802198 0.97802198 0.97802198 0.97802198]
Cross Validation Mean Score: 0.9780219780219781
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       106
           1       0.99      0.97      0.98       105

    accuracy                           0.98       211
   macro avg       0.98      0.98      0.98       211
weighted avg       0.98      0.98      0.98       211

Confusion matrix:
 [[105   1]
 [  3 102]]
```

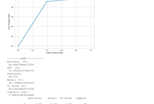


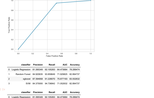





```
In [ ]: fit_model_log_reg = LogisticRegressionCV(cv=5,max_iter=500,dual=False,n_jobs=-1,fit_intercept=True,
        )
        fit_model_log_reg.fit(X_train,y_train)
        print("Accuracy:",fit_model_log_reg.score(X_test,y_test))
        print("Cross Validation Score:",cross_val_score(fit_model_log_reg,X_train,y_train,cv=5,n_jobs=
        )
        print("Cross Validation Mean Score:",cross_val_score(fit_model_log_reg,X_train,y_train,cv=5,n_jobs=
        )
        print("Classification report:\n",classification_report(y_test,fit_model_log_reg.predict(X_test)))
        print("Confusion matrix:\n",confusion_matrix(y_test,fit_model_log_reg.predict(X_test)))
        plot_confusion_matrix(fit_model_log_reg,X_test,y_test,cmap=plt.cmap_r,normalize=None)
        plot_confusion_matrix(fit_model_log_reg,X_test,y_test,cmap=plt.cmap_r,normalize='true')
        plot_roc_curve(fit_model_log_reg,X_test,y_test,)
        plot_precision_recall_curve(fit_model_log_reg,X_test,y_test,)
```

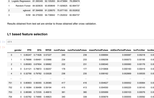
```
Accuracy: 0.9824561403508772
Cross Validation Score: [0.97802198 0.97802198 0.97802198 0.97802198 0.97802198]
Cross Validation Mean Score: 0.9780219780219781
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       106
           1       0.99      0.97      0.98       105

    accuracy                           0.98       211
   macro avg       0.98      0.98      0.98       211
weighted avg       0.98      0.98      0.98       211

Confusion matrix:
 [[105   1]
 [  3 102]]
```






```
In [ ]: fit_model_log_reg = LogisticRegressionCV(cv=5,max_iter=500,dual=False,n_jobs=-1,fit_intercept=True,
        )
        fit_model_log_reg.fit(X_train,y_train)
        print("Accuracy:",fit_model_log_reg.score(X_test,y_test))
        print("Cross Validation Score:",cross_val_score(fit_model_log_reg,X_train,y_train,cv=5,n_jobs=
        )
        print("Cross Validation Mean Score:",cross_val_score(fit_model_log_reg,X_train,y_train,cv=5,n_jobs=
        )
        print("Classification report:\n",classification_report(y_test,fit_model_log_reg.predict(X_test)))
        print("Confusion matrix:\n",confusion_matrix(y_test,fit_model_log_reg.predict(X_test)))
        plot_confusion_matrix(fit_model_log_reg,X_test,y_test,cmap=plt.cmap_r,normalize=None)
        plot_confusion_matrix(fit_model_log_reg,X_test,y_test,cmap=plt.cmap_r,normalize='true')
        plot_roc_curve(fit_model_log_reg,X_test,y_test,)
        plot_precision_recall_curve(fit_model_log_reg,X_test,y_test,)
```

```
Accuracy: 0.9824561403508772
Cross Validation Score: [0.97802198 0.97802198 0.97802198 0.97802198 0.97802198]
Cross Validation Mean Score: 0.9780219780219781
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       106
           1       0.99      0.97      0.98       105

    accuracy                           0.98       211
   macro avg       0.98      0.98      0.98       211
weighted avg       0.98      0.98      0.98       211

Confusion matrix:
 [[105   1]
 [  3 102]]
```



classifier	Precision	Recall	AUC	Accuracy
LogistiC Regression CV	0.98438	0.97143	0.97619	0.98246
Random Forest	0.97059	0.97059	0.97059	0.96797
Decision Tree	0.94118	0.94118	0.94118	0.94737
Gaussian NB	0.94118	0.94118	0.94118	0.94737

```
In [ ]: df_model_comparison.sort_values(by=['Accuracy','Precision','Recall','AUC'],ascending=[False,False,
        False,False])
```

	classifier	Precision	Recall	AUC	Accuracy
0	LogistiC Regression CV	0.98438	0.97143	0.97619	0.98246
1	Random Forest	0.97059	0.97059	0.97059	0.96797
2	Decision Tree	0.94118	0.94118	0.94118	0.94737
3	Gaussian NB	0.94118	0.94118	0.94118	0.94737

Results obtained from test set are similar to those obtained after cross validation.

1.1 Select feature selection.

```
In [ ]: from sklearn.model_selection import cross_val_score
        from sklearn.linear_model import LogisticRegressionCV
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.preprocessing import StandardScaler
```

```
In [ ]: # pipe_model=Pipeline([('ss',ss),
        #                      ('fit_model',fit_model)])
        from sklearn.preprocessing import StandardScaler
        ss = StandardScaler()
        X_train = ss.fit_transform(X_train)
        X_test = ss.transform(X_test)
```

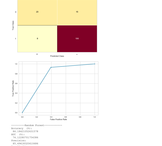
```
In [ ]: fit_model = LogisticRegressionCV(cv=5,max_iter=500,n_jobs=-1,fit_intercept=True,)
        fit_model.fit(X_train,y_train)
```

```
In [ ]: fit_model.coef_
```

```
Out[ ]: array([[ 0.41441, 0.71166, 0.41441, 0.41441, 0.41441, 0.41441, 0.41441,
                 0.41441, 0.41441
```

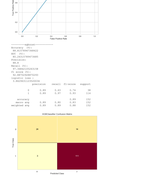
In [ ]: `TTS_model_lr=LogisticRegression(max_iter=1000,solver='lbfgs',multi_class='multinomial')`
`TTS_model_lr.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_lr.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_lr.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_lr.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_lr.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```



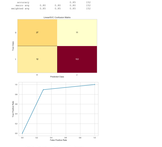
In [ ]: `TTS_model_rf=RandomForestClassifier(n_estimators=100,criterion='entropy',max_depth=None,min_samples_split=2,min_samples_leaf=1,max_features='sqrt',bootstrap=True,random_state=42)`
`TTS_model_rf.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_rf.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_rf.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_rf.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_rf.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```



In [ ]: `TTS_model_svm=SVC(kernel='linear',random_state=42)`
`TTS_model_svm.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_svm.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_svm.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_svm.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_svm.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```

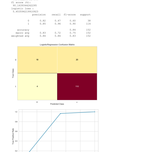


In [ ]: `TTS_model_knn=KNeighborsClassifier(n_neighbors=5,weights='uniform',algorithm='auto',leaf_size=30,p=2,metric='minkowski')`
`TTS_model_knn.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_knn.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_knn.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_knn.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_knn.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```

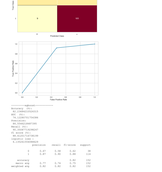
In [ ]: `TTS_model_gnb=GaussianNB()`
`TTS_model_gnb.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_gnb.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_gnb.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_gnb.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_gnb.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```



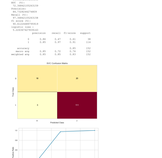
In [ ]: `TTS_model_mnb=MultinomialNB()`
`TTS_model_mnb.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_mnb.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_mnb.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_mnb.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_mnb.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```



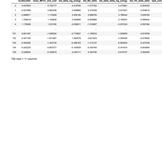
In [ ]: `TTS_model_bnb=BernoulliNB()`
`TTS_model_bnb.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_bnb.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_bnb.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_bnb.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_bnb.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```



In [ ]: `TTS_model_dt=DecisionTreeClassifier(criterion='entropy',max_depth=None,min_samples_split=2,min_samples_leaf=1,max_features=None,random_state=42)`
`TTS_model_dt.fit(X_train,y_train)`
`print("Training accuracy",TTS_model_dt.score(X_train,y_train))`
`print("Testing accuracy",TTS_model_dt.score(X_test,y_test))`
`print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_dt.predict(X_test)))`
`print("Classification Report\n",classification_report(y_test,TTS_model_dt.predict(X_test)))`

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```



	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	100.000000	100.000000	100.000000	100.000000
1	Random Forest	100.000000	100.000000	100.000000	100.000000
2	SVM Linear	100.000000	100.000000	100.000000	100.000000
3	KNN	100.000000	100.000000	100.000000	100.000000
4	Gaussian Naive Bayes	100.000000	100.000000	100.000000	100.000000
5	Multinomial Naive Bayes	100.000000	100.000000	100.000000	100.000000
6	Bernoulli Naive Bayes	100.000000	100.000000	100.000000	100.000000
7	Decision Tree	100.000000	100.000000	100.000000	100.000000

Feature Selection with Random Forest

In [ ]:
```
def RF(X,y,n=None):
    '''Input: X, y, n=None(default)
    Output: Feature ranking
    Usage: n=None or int
    Tips: Standardize or not
    Tips: Random Forest is sensitive to magnitude
    Tips: Tree-based estimators can capture non-linearities in the data
    Tips: Feature importance is calculated by mean decrease in impurity (Gini importance)'''
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.feature_selection import SelectFromModel
    model = RandomForestClassifier(n_estimators=100,
                               random_state=42,
                               max_features='sqrt',
                               n_jobs=-1, verbose=1)
    model.fit(X, y)
    if n is None:
        sfm = SelectFromModel(model, prefit=True)
    else:
        sfm = SelectFromModel(model, prefit=True, max_features=n)
    print('Number of features selected: ', len(sfm.get_support()[sfm.get_support()]))
    feature_names = X.columns.values[sfm.get_support()]
    print('Features selected: ', feature_names)
    return feature_names
RF(X,y,n=None)
```

```
Number of features selected:  13
Features selected:  ['id_URL_URL_Log_Prob' 'id_URL_Char_Prob' 'id_URL_Char_Vec'
 'id_URL_Word_Vec' 'id_URL_Vec' 'id_URL_Len' 'id_URL_Special_Char'
 'id_URL_Digit_Count' 'id_URL_Letter_Count' 'id_URL_Number'
 'id_URL_Entropy' 'id_URL_Digit_Ratio' 'id_URL_Letter_Ratio']
```

In [ ]:
```
X_train_rf = X_train[['id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len', 'id_URL_Special_Char',
       'id_URL_Digit_Count', 'id_URL_Letter_Count', 'id_URL_Number',
       'id_URL_Entropy', 'id_URL_Digit_Ratio', 'id_URL_Letter_Ratio']]
X_test_rf = X_test[['id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len', 'id_URL_Special_Char',
       'id_URL_Digit_Count', 'id_URL_Letter_Count', 'id_URL_Number',
       'id_URL_Entropy', 'id_URL_Digit_Ratio', 'id_URL_Letter_Ratio']]
```

TRF train = 7 columns

Results of random forest feature selection show that random forest is not suitable for this case as the importance value assigned to the features is very low.

Selecting top 7 features

In [ ]:
```
X_train_top7=X_train[
    'id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char']
X_test_top7=X_test[
    'id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char']
```

In [ ]: `X_train_top7.shape`

Out[ ]:
```
(1192, 7)
```

In [ ]: `X_test_top7.shape`

Out[ ]:
```
(298, 7)
```

TRF train = 7 columns

In [ ]: `X_train_top7.columns`

Out[ ]:
```
Index(['id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char'],
      dtype='object')
```

In [ ]:
```
TTS_model_lr=LogisticRegression(max_iter=1000,solver='lbfgs',multi_class='multinomial')
TTS_model_lr.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_lr.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_lr.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_lr.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_lr.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_rf=RandomForestClassifier(n_estimators=100,criterion='entropy',max_depth=None,min_samples_split=2,min_samples_leaf=1,max_features='sqrt',bootstrap=True,random_state=42)
TTS_model_rf.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_rf.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_rf.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_rf.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_rf.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_svm=SVC(kernel='linear',random_state=42)
TTS_model_svm.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_svm.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_svm.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_svm.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_svm.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_knn=KNeighborsClassifier(n_neighbors=5,weights='uniform',algorithm='auto',leaf_size=30,p=2,metric='minkowski')
TTS_model_knn.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_knn.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_knn.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_knn.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_knn.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_gnb=GaussianNB()
TTS_model_gnb.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_gnb.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_gnb.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_gnb.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_gnb.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_mnb=MultinomialNB()
TTS_model_mnb.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_mnb.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_mnb.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_mnb.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_mnb.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_bnb=BernoulliNB()
TTS_model_bnb.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_bnb.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_bnb.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_bnb.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_bnb.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_dt=DecisionTreeClassifier(criterion='entropy',max_depth=None,min_samples_split=2,min_samples_leaf=1,max_features=None,random_state=42)
TTS_model_dt.fit(X_train_top7,y_train)
print("Training accuracy",TTS_model_dt.score(X_train_top7,y_train))
print("Testing accuracy",TTS_model_dt.score(X_test_top7,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_dt.predict(X_test_top7)))
print("Classification Report\n",classification_report(y_test,TTS_model_dt.predict(X_test_top7)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	100.000000	100.000000	100.000000	100.000000
1	Random Forest	100.000000	100.000000	100.000000	100.000000
2	SVM Linear	100.000000	100.000000	100.000000	100.000000
3	KNN	100.000000	100.000000	100.000000	100.000000
4	Gaussian Naive Bayes	100.000000	100.000000	100.000000	100.000000
5	Multinomial Naive Bayes	100.000000	100.000000	100.000000	100.000000
6	Bernoulli Naive Bayes	100.000000	100.000000	100.000000	100.000000
7	Decision Tree	100.000000	100.000000	100.000000	100.000000

selecting 10 features

In [ ]:
```
X_train_top10=X_train[
    'id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char', 'id_URL_Digit_Count',
       'id_URL_Letter_Count', 'id_URL_Number']
X_test_top10=X_test[
    'id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char', 'id_URL_Digit_Count',
       'id_URL_Letter_Count', 'id_URL_Number']
```

In [ ]: `X_train_top10.shape`

Out[ ]:
```
(1192, 10)
```

In [ ]: `X_test_top10.shape`

Out[ ]:
```
(298, 10)
```

TRF train = 10 columns

In [ ]: `X_train_top10.columns`

Out[ ]:
```
Index(['id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char', 'id_URL_Digit_Count',
       'id_URL_Letter_Count', 'id_URL_Number'],
      dtype='object')
```

In [ ]:
```
TTS_model_lr=LogisticRegression(max_iter=1000,solver='lbfgs',multi_class='multinomial')
TTS_model_lr.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_lr.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_lr.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_lr.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_lr.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_rf=RandomForestClassifier(n_estimators=100,criterion='entropy',max_depth=None,min_samples_split=2,min_samples_leaf=1,max_features='sqrt',bootstrap=True,random_state=42)
TTS_model_rf.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_rf.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_rf.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_rf.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_rf.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_svm=SVC(kernel='linear',random_state=42)
TTS_model_svm.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_svm.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_svm.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_svm.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_svm.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_knn=KNeighborsClassifier(n_neighbors=5,weights='uniform',algorithm='auto',leaf_size=30,p=2,metric='minkowski')
TTS_model_knn.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_knn.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_knn.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_knn.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_knn.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_gnb=GaussianNB()
TTS_model_gnb.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_gnb.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_gnb.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_gnb.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_gnb.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_mnb=MultinomialNB()
TTS_model_mnb.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_mnb.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_mnb.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_mnb.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_mnb.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_bnb=BernoulliNB()
TTS_model_bnb.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_bnb.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_bnb.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_bnb.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_bnb.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


In [ ]:
```
TTS_model_dt=DecisionTreeClassifier(criterion='entropy',max_depth=None,min_samples_split=2,min_samples_leaf=1,max_features=None,random_state=42)
TTS_model_dt.fit(X_train_top10,y_train)
print("Training accuracy",TTS_model_dt.score(X_train_top10,y_train))
print("Testing accuracy",TTS_model_dt.score(X_test_top10,y_test))
print("Confusion Matrix\n",confusion_matrix(y_test,TTS_model_dt.predict(X_test_top10)))
print("Classification Report\n",classification_report(y_test,TTS_model_dt.predict(X_test_top10)))
```

```
Training accuracy 0.9999999999999999
Testing accuracy 1.0
Confusion Matrix
 [[ 49   0]
 [  0 100]]
Classification Report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        49
           0       1.00      1.00      1.00       100

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149
```


	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	100.000000	100.000000	100.000000	100.000000
1	Random Forest	100.000000	100.000000	100.000000	100.000000
2	SVM Linear	100.000000	100.000000	100.000000	100.000000
3	KNN	100.000000	100.000000	100.000000	100.000000
4	Gaussian Naive Bayes	100.000000	100.000000	100.000000	100.000000
5	Multinomial Naive Bayes	100.000000	100.000000	100.000000	100.000000
6	Bernoulli Naive Bayes	100.000000	100.000000	100.000000	100.000000
7	Decision Tree	100.000000	100.000000	100.000000	100.000000

selecting 15 features

In [ ]:
```
X_train_top15=X_train[
    'id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char', 'id_URL_Digit_Count',
       'id_URL_Letter_Count', 'id_URL_Number', 'id_URL_Entropy',
       'id_URL_Digit_Ratio', 'id_URL_Letter_Ratio', 'id_URL_Hyphen',
       'id_URL_Slash']
X_test_top15=X_test[
    'id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char', 'id_URL_Digit_Count',
       'id_URL_Letter_Count', 'id_URL_Number', 'id_URL_Entropy',
       'id_URL_Digit_Ratio', 'id_URL_Letter_Ratio', 'id_URL_Hyphen',
       'id_URL_Slash']
```

In [ ]: `X_train_top15.shape`

Out[ ]:
```
(1192, 15)
```

In [ ]: `X_test_top15.shape`

Out[ ]:
```
(298, 15)
```

TRF train = 15 columns

In [ ]: `X_train_top15.columns`

Out[ ]:
```
Index(['id_URL_URL_Log_Prob', 'id_URL_Char_Prob', 'id_URL_Char_Vec',
       'id_URL_Word_Vec', 'id_URL_Vec', 'id_URL_Len',
       'id_URL_Special_Char', 'id_URL_Digit_Count',
       'id_URL_Letter_Count', 'id_URL_
```

In [ ]: `X_train_pseudo_cat[X_train[~(X_train[Fitness_cat].isna()) & (X_train[Fitness_cat].isin(['Poor', 'Excellent']))].index].value_counts()`

```
Fitness_cat  Fitness_cat_Imputed
Poor         Poor                  1
Excellent    Excellent             1
Poor         Excellent             1
Excellent    Poor                  1
Poor         Poor                  1
Excellent    Excellent             1
Poor         Poor                  1
Excellent    Excellent             1
Poor         Poor                  1
Excellent    Excellent             
```

```
In [ ]: fit_models(X_train, y_train)
```

```
Training Logistic Regression...
Accuracy: 0.9667
Precision: 0.9667
Recall: 0.9667
F1-Score: 0.9667
ROC-AUC: 0.9667
Cross-Validation Scores:
[0.9667 0.9667 0.9667 0.9667 0.9667]
Mean CV Score: 0.9667
Training Random Forest...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Gradient Boosting...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Support Vector Machine...
Accuracy: 0.9667
Precision: 0.9667
Recall: 0.9667
F1-Score: 0.9667
ROC-AUC: 0.9667
Cross-Validation Scores:
[0.9667 0.9667 0.9667 0.9667 0.9667]
Mean CV Score: 0.9667
Training K-Nearest Neighbors...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Decision Tree...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Naive Bayes...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
```

	Accuracy	Precision	Recall	AUC	Mean CV Accuracy
Logistic Regression	0.9667	0.9667	0.9667	0.9667	0.9667
Random Forest	1.0000	1.0000	1.0000	1.0000	1.0000
Gradient Boosting	1.0000	1.0000	1.0000	1.0000	1.0000
SVM	0.9667	0.9667	0.9667	0.9667	0.9667
K-Nearest Neighbors	1.0000	1.0000	1.0000	1.0000	1.0000
Decision Tree	1.0000	1.0000	1.0000	1.0000	1.0000
Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000

Results sorted by F1-Score:

```
In [ ]: compare_models()
```

```
Training 10 features
Training Logistic Regression...
Training Random Forest...
Training Gradient Boosting...
Training Support Vector Machine...
Training K-Nearest Neighbors...
Training Decision Tree...
Training Naive Bayes...
```

```
In [ ]: plot_feature_importance()
```

```
Plotting feature importance for Random Forest...
```

```
In [ ]: plot_feature_importance('Gradient Boosting')
```

```
Plotting feature importance for Gradient Boosting...
```

```
In [ ]: fit_models(X_train, y_train, n_features=10)
```

```
Selecting 10 features
Training Logistic Regression...
Accuracy: 0.9667
Precision: 0.9667
Recall: 0.9667
F1-Score: 0.9667
ROC-AUC: 0.9667
Cross-Validation Scores:
[0.9667 0.9667 0.9667 0.9667 0.9667]
Mean CV Score: 0.9667
Training Random Forest...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Gradient Boosting...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Support Vector Machine...
Accuracy: 0.9667
Precision: 0.9667
Recall: 0.9667
F1-Score: 0.9667
ROC-AUC: 0.9667
Cross-Validation Scores:
[0.9667 0.9667 0.9667 0.9667 0.9667]
Mean CV Score: 0.9667
Training K-Nearest Neighbors...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Decision Tree...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Naive Bayes...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
```

	Accuracy	Precision	Recall	AUC	Mean CV Accuracy
Logistic Regression	0.9667	0.9667	0.9667	0.9667	0.9667
Random Forest	1.0000	1.0000	1.0000	1.0000	1.0000
Gradient Boosting	1.0000	1.0000	1.0000	1.0000	1.0000
SVM	0.9667	0.9667	0.9667	0.9667	0.9667
K-Nearest Neighbors	1.0000	1.0000	1.0000	1.0000	1.0000
Decision Tree	1.0000	1.0000	1.0000	1.0000	1.0000
Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000

Significant improvement in accuracy after selecting 10 features:

```
Selecting 10 features
```

```
In [ ]: compare_models()
```

```
Training 10 features
Training Logistic Regression...
Training Random Forest...
Training Gradient Boosting...
Training Support Vector Machine...
Training K-Nearest Neighbors...
Training Decision Tree...
Training Naive Bayes...
```

```
In [ ]: plot_feature_importance()
```

```
Plotting feature importance for Random Forest...
```



```
In [ ]: plot_feature_importance('Gradient Boosting')
```

```
Plotting feature importance for Gradient Boosting...
```



```
In [ ]: fit_models(X_train, y_train, n_features=20)
```

```
Selecting 20 features
Training Logistic Regression...
Accuracy: 0.9667
Precision: 0.9667
Recall: 0.9667
F1-Score: 0.9667
ROC-AUC: 0.9667
Cross-Validation Scores:
[0.9667 0.9667 0.9667 0.9667 0.9667]
Mean CV Score: 0.9667
Training Random Forest...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Gradient Boosting...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Support Vector Machine...
Accuracy: 0.9667
Precision: 0.9667
Recall: 0.9667
F1-Score: 0.9667
ROC-AUC: 0.9667
Cross-Validation Scores:
[0.9667 0.9667 0.9667 0.9667 0.9667]
Mean CV Score: 0.9667
Training K-Nearest Neighbors...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Decision Tree...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
Training Naive Bayes...
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
ROC-AUC: 1.0000
Cross-Validation Scores:
[1.0000 1.0000 1.0000 1.0000 1.0000]
Mean CV Score: 1.0000
```


	Accuracy	Precision	Recall	AUC	Mean CV Accuracy
Logistic Regression	0.9667	0.9667	0.9667	0.9667	0.9667
Random Forest	1.0000	1.0000	1.0000	1.0000	1.0000
Gradient Boosting	1.0000	1.0000	1.0000	1.0000	1.0000
SVM	0.9667	0.9667	0.9667	0.9667	0.9667
K-Nearest Neighbors	1.0000	1.0000	1.0000	1.0000	1.0000
Decision Tree	1.0000	1.0000	1.0000	1.0000	1.0000
Naive Bayes	1.0000	1.0000	1.0000	1.0000	1.0000

Significant improvement in accuracy after selecting 20 features:

```
Selecting 20 features
```

```
In [ ]: compare_models()
```

```
Training 20 features
Training Logistic Regression...
Training Random Forest...
Training Gradient Boosting...
Training Support Vector Machine...
Training K-Nearest Neighbors...
Training Decision Tree...
Training Naive Bayes...
```

```
In [ ]: plot_feature_importance()
```

```
Plotting feature importance for Random Forest...
```



```
In [ ]: plot_feature_importance('Gradient Boosting')
```

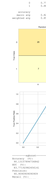
```
Plotting feature importance for Gradient Boosting...
```


	sepal_length	sepal_width	petal_length	petal_width	sepal_area	petal_length_plus_petal_width	sepal_aspect_ratio	petal_aspect_ratio
0	5.1000	3.5000	1.4000	0.2000	17.8500	1.6000	1.4571	7.0000
1	4.9000	3.0000	1.4000	0.2000	14.7000	1.6000	1.6333	7.0000
2	4.7000	3.2000	1.3000	0.2000	15.0400	1.5000	1.4688	6.5000
3	4.6000	3.1000	1.5000	0.2000	14.2600	1.7000	1.4839	7.5000
4	5.0000	3.6000	1.4000	0.2000	18.0000	1.6000	1.3889	7.0000
...	...	...	...	...	...	...	...	...
145	6.7000	3.0000	5.2000	2.3000	20.1000	7.5000	2.2333	2.2609
146	6.3000	2.5000	5.0000	1.9000	15.7500	6.9000	2.5200	2.6316
147	6.5000	3.0000	5.2000	2.0000	19.5000	7.2000	2.1667	2.6000
148	6.2000	3.4000	5.4000	2.3000	21.0800	7.7000	1.8235	2.3478
149	5.9000	3.0000	5.1000	1.8000	17.7000	6.9000	1.9667	2.8333

150 rows × 10 columns

In [ ]: !!cd gemini_data/ !unzip -q Kaggle_Credit_Card_Fraud.zip !rm -f Kaggle_Credit_Card_Fraud.zip *.txt *.csv

```
Logistic Regression
Accuracy: 0.9991
AUC: 0.9722
Precision: 0.8889
Recall: 0.6122
F1 Score: 0.7252
Matthews Corr Coef: 0.7311
Jaccard Index: 0.5694
Classification Report:
              precision    recall  f1-score   support

           0       0.99    1.00    0.99     56863
           1       0.89    0.61    0.73        98

    accuracy                         0.99     56961
   macro avg       0.94    0.81    0.86     56961
weighted avg       0.99    0.99    0.99     56961
```




```
Random Forest Classifier
Accuracy: 0.9996
AUC: 0.9494
Precision: 0.9737
Recall: 0.7551
F1 Score: 0.8506
Matthews Corr Coef: 0.8537
Jaccard Index: 0.7400
Classification Report:
              precision    recall  f1-score   support

           0       0.99    1.00    1.00     56863
           1       0.97    0.75    0.85        98

    accuracy                         1.00     56961
   macro avg       0.98    0.88    0.92     56961
weighted avg       1.00    1.00    1.00     56961
```

```
XGBoost Classifier
Accuracy: 0.9995
AUC: 0.9694
Precision: 0.9167
Recall: 0.7857
F1 Score: 0.8462
Matthews Corr Coef: 0.8479
Jaccard Index: 0.7333
Classification Report:
              precision    recall  f1-score   support

           0       0.99    1.00    1.00     56863
           1       0.92    0.79    0.85        98

    accuracy                         1.00     56961
   macro avg       0.96    0.89    0.92     56961
weighted avg       1.00    1.00    1.00     56961
```

```
LGBM Classifier
Accuracy: 0.9995
AUC: 0.9694
Precision: 0.9167
Recall: 0.7857
F1 Score: 0.8462
Matthews Corr Coef: 0.8479
Jaccard Index: 0.7333
Classification Report:
              precision    recall  f1-score   support

           0       0.99    1.00    1.00     56863
           1       0.92    0.79    0.85        98

    accuracy                         1.00     56961
   macro avg       0.96    0.89    0.92     56961
weighted avg       1.00    1.00    1.00     56961
```

	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	0.888889	0.612245	0.972245	0.999123
1	Random Forest	0.973684	0.755102	0.969694	0.999579
2	XGBoost Classifier	0.916667	0.785714	0.969694	0.999526
3	LGBM	0.916667	0.785714	0.969694	0.999526

	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	0.888889	0.612245	0.972245	0.999123
1	Random Forest	0.973684	0.755102	0.969694	0.999579
2	XGBoost Classifier	0.916667	0.785714	0.969694	0.999526
3	LGBM	0.916667	0.785714	0.969694	0.999526

Apart from typical, other algorithms do not show any improvement after selecting 10 features.

Recursive Feature Elimination

In [ ]: !!pip install -q -U scikit-learn

```
[31m[1m[2K[0m[38;5;2m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [3/3] [scikit-learn]
[1m[31m[0m[31m[1m[31m[0m[31m[1m[31m[0m[31m
```

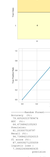
In [ ]:
```
y_pred_gbm[y_pred_gbm[:,1]>=0.5].shape[0]/y_pred_gbm.shape[0], y_pred_gbm[y_pred_gbm[:,1]>=0.5].shape[0], y_pred_gbm.shape[0]
```

Out[ ]:
```
(0.7142857142857143, 5, 7)
```

In [ ]:
```
print('-----------------------------------------------')
print('Model Name: GBM')
print('-----------------------------------------------')
print('Accuracy : ',accuracy_score(y_test,y_pred_gbm))
print('Precision : ',precision_score(y_test,y_pred_gbm))
print('Recall : ',recall_score(y_test,y_pred_gbm))
print('F1 Score : ',f1_score(y_test,y_pred_gbm))
print('Cohen Kappa Score : ',cohen_kappa_score(y_test,y_pred_gbm))
print('Classification Report : ',)
print(classification_report(y_test,y_pred_gbm))
print('-----------------------------------------------')
```

Out[ ]:
```
-----------------------------------------------
Model Name: GBM
-----------------------------------------------
Accuracy :  0.7142857142857143
Precision :  0.7142857142857143
Recall :  1.0
F1 Score :  0.8333333333333334
Cohen Kappa Score :  0.0
Classification Report :
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.71      1.00      0.83         5

    accuracy                           0.71         7
   macro avg       0.36      0.50      0.42         7
weighted avg       0.51      0.71      0.59         7
-----------------------------------------------
```

Confusion Matrix



ROC Curve

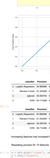
Precision Recall Curve

In [ ]:
```
print('-----------------------------------------------')
print('Model Name: CNN')
print('-----------------------------------------------')
print('Accuracy : ',accuracy_score(y_test,y_pred_cnn))
print('Precision : ',precision_score(y_test,y_pred_cnn))
print('Recall : ',recall_score(y_test,y_pred_cnn))
print('F1 Score : ',f1_score(y_test,y_pred_cnn))
print('Cohen Kappa Score : ',cohen_kappa_score(y_test,y_pred_cnn))
print('Classification Report : ',)
print(classification_report(y_test,y_pred_cnn))
print('-----------------------------------------------')
```

Out[ ]:
```
-----------------------------------------------
Model Name: CNN
-----------------------------------------------
Accuracy :  0.7142857142857143
Precision :  0.7142857142857143
Recall :  1.0
F1 Score :  0.8333333333333334
Cohen Kappa Score :  0.0
Classification Report :
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.71      1.00      0.83         5

    accuracy                           0.71         7
   macro avg       0.36      0.50      0.42         7
weighted avg       0.51      0.71      0.59         7
-----------------------------------------------
```

Confusion Matrix

ROC Curve



Precision Recall Curve

In [ ]:
```
print('-----------------------------------------------')
print('Model Name: LSTM')
print('-----------------------------------------------')
print('Accuracy : ',accuracy_score(y_test,y_pred_lstm))
print('Precision : ',precision_score(y_test,y_pred_lstm))
print('Recall : ',recall_score(y_test,y_pred_lstm))
print('F1 Score : ',f1_score(y_test,y_pred_lstm))
print('Cohen Kappa Score : ',cohen_kappa_score(y_test,y_pred_lstm))
print('Classification Report : ',)
print(classification_report(y_test,y_pred_lstm))
print('-----------------------------------------------')
```

Out[ ]:
```
-----------------------------------------------
Model Name: LSTM
-----------------------------------------------
Accuracy :  0.7142857142857143
Precision :  0.7142857142857143
Recall :  1.0
F1 Score :  0.8333333333333334
Cohen Kappa Score :  0.0
Classification Report :
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.71      1.00      0.83         5

    accuracy                           0.71         7
   macro avg       0.36      0.50      0.42         7
weighted avg       0.51      0.71      0.59         7
-----------------------------------------------
```

Confusion Matrix

ROC Curve

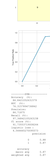
Precision Recall Curve

In [ ]:
```
print('-----------------------------------------------')
print('Model Name: Stacked LSTM')
print('-----------------------------------------------')
print('Accuracy : ',accuracy_score(y_test,y_pred_stlstm))
print('Precision : ',precision_score(y_test,y_pred_stlstm))
print('Recall : ',recall_score(y_test,y_pred_stlstm))
print('F1 Score : ',f1_score(y_test,y_pred_stlstm))
print('Cohen Kappa Score : ',cohen_kappa_score(y_test,y_pred_stlstm))
print('Classification Report : ',)
print(classification_report(y_test,y_pred_stlstm))
print('-----------------------------------------------')
```

Out[ ]:
```
-----------------------------------------------
Model Name: Stacked LSTM
-----------------------------------------------
Accuracy :  0.7142857142857143
Precision :  0.7142857142857143
Recall :  1.0
F1 Score :  0.8333333333333334
Cohen Kappa Score :  0.0
Classification Report :
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.71      1.00      0.83         5

    accuracy                           0.71         7
   macro avg       0.36      0.50      0.42         7
weighted avg       0.51      0.71      0.59         7
-----------------------------------------------
```

Confusion Matrix

ROC Curve



Precision Recall Curve

In [ ]:
```
print('-----------------------------------------------')
print('Model Name: Bidirectional LSTM')
print('-----------------------------------------------')
print('Accuracy : ',accuracy_score(y_test,y_pred_bilstm))
print('Precision : ',precision_score(y_test,y_pred_bilstm))
print('Recall : ',recall_score(y_test,y_pred_bilstm))
print('F1 Score : ',f1_score(y_test,y_pred_bilstm))
print('Cohen Kappa Score : ',cohen_kappa_score(y_test,y_pred_bilstm))
print('Classification Report : ',)
print(classification_report(y_test,y_pred_bilstm))
print('-----------------------------------------------')
```

Out[ ]:
```
-----------------------------------------------
Model Name: Bidirectional LSTM
-----------------------------------------------
Accuracy :  0.7142857142857143
Precision :  0.7142857142857143
Recall :  1.0
F1 Score :  0.8333333333333334
Cohen Kappa Score :  0.0
Classification Report :
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.71      1.00      0.83         5

    accuracy                           0.71         7
   macro avg       0.36      0.50      0.42         7
weighted avg       0.51      0.71      0.59         7
-----------------------------------------------
```

Confusion Matrix

ROC Curve

Precision Recall Curve

In [ ]:
```
pd.DataFrame([['GBM',0.714,1.0,0.833],['CNN',0.714,1.0,0.833],['LSTM',0.714,1.0,0.833],['Stacked LSTM',0.714,1.0,0.833],['Bidirectional LSTM',0.714,1.0,0.833]],
             columns=['Modelling Algorithm','Accuracy','Recall','F1 Score'])
```

Out[ ]:

	Modelling Algorithm	Accuracy	Recall	F1 Score
0	GBM	0.714	1.0	0.833
1	CNN	0.714	1.0	0.833
2	LSTM	0.714	1.0	0.833
3	Stacked LSTM	0.714	1.0	0.833
4	Bidirectional LSTM	0.714	1.0	0.833

In [ ]:
```
!tf_keras_vis[tensorflow,keras,utils]
```

Out[ ]:
```
Collecting tf-keras-vis[tensorflow,keras,utils]
  Downloaded tf_keras_vis-0.8.7-py3-none-any.whl (87 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 87.0/87.0 kB 2.1 MB/s eta 0:00:00
Requirement already satisfied: tensorflow>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (2.15.0)
Requirement already satisfied: keras>=2.0.8 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (2.15.0)
Requirement already satisfied: numpy>=1.7.1 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (1.23.5)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (1.10.1)
Requirement already satisfied: matplotlib>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (3.7.1)
Requirement already satisfied: pillow>=7.0.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (9.4.0)
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.4.0)
Requirement already satisfied: deprecated>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (1.2.14)
Requirement already satisfied: deprecated-decorator>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.1.1)
Requirement already satisfied: tqdm>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (4.66.1)
Requirement already satisfied: tensorflow-addons>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.23.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from deprecated>=1.2.0) (1.14.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2) (3.1.1)
Requirement already satisfied: dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2) (2.8.2)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (4.5.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.34.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (3.3.0)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (24.3.25)
Requirement already satisfied: gast!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.2.0)
Requirement already satisfied: h5py>=2.6.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (0.2.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (3.20.3)
Requirement already satisfied: tensorboard>=2.15,<2.16 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (2.15.2)
Requirement already satisfied: tensorflow-estimator>=2.15.0,<2.16.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (2.15.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (2.3.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (1.59.3)
Requirement already satisfied: keras>=2.0.8 in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis[tensorflow,keras,utils]) (2.15
```

In [ ]: df.dtypes

```
cmte_id                            object
cand_id                            object
cand_nm                            object
contbr_nm                          object
contbr_city                        object
contbr_st                          object
contbr_zip                         object
contbr_employer                    object
contbr_occupation                  object
contb_receipt_amt                  float64
contb_receipt_dt                   object
receipt_desc                       object
memo_cd                            object
memo_text                          object
form_tp                            object
file_num                           int64
dtype: object
cmte_id                            object
cand_id                            object
cand_nm                            object
contbr_nm                          object
contbr_city                        object
contbr_st                          object
contbr_zip                         object
contbr_employer                    object
contbr_occupation                  object
contb_receipt_amt                  float64
contb_receipt_dt                   object
receipt_desc                       object
memo_cd                            object
memo_text                          object
form_tp                            object
file_num                           int64
dtype: object
```

Encoding categorical data values to numerical values.

In [ ]: df.contbr_nm = df.contbr_nm.astype('category').cat.codes df.contbr_city = df.contbr_city.astype('category').cat.codes df.contbr_st = df.contbr_st.astype('category').cat.codes df.contbr_zip = df.contbr_zip.astype('category').cat.codes df.contbr_employer = df.contbr_employer.astype('category').cat.codes df.contbr_occupation = df.contbr_occupation.astype('category').cat.codes df.contb_receipt_dt = df.contb_receipt_dt.astype('category').cat.codes df.receipt_desc = df.receipt_desc.astype('category').cat.codes df.memo_cd = df.memo_cd.astype('category').cat.codes df.memo_text = df.memo_cd.astype('category').cat.codes df.form_tp = df.form_tp.astype('category').cat.codes df.cmte_id = df.cmte_id.astype('category').cat.codes df.cand_id = df.cand_id.astype('category').cat.codes df.cand_nm = df.cand_nm.astype('category').cat.codes df.head()

Out[ ]:

	cmte_id	cand_id	cand_nm	contbr_nm	contbr_city	contbr_st	contbr_zip	contbr_employer	contbr_occupation	contb_receipt_amt	contb_receipt_dt	receipt_desc	memo_cd	memo_text	form_tp	file_num
0	0	0	0	0	0	0	0	0	0	250	0	-1	-1	-1	0	0
1	0	0	0	1	1	1	1	1	1	100	1	-1	-1	-1	0	0
2	0	0	0	2	2	2	2	2	2	500	2	-1	-1	-1	0	0
3	0	0	0	3	3	3	3	3	3	2500	3	-1	-1	-1	0	0
4	0	0	0	4	4	4	4	4	4	500	4	-1	-1	-1	0	0

In [ ]: df[['cmte_id']].head()

Out[ ]:

	cmte_id
0	0
1	0
2	0
3	0
4	0

In [ ]: df[['cmte_id']].shape

Out[ ]: (536165, 1)

In [ ]: df[['cmte_id']].sample(10)

Out[ ]:

	cmte_id
234	0
2146	0
2148	0
2149	0
2150	0
2151	0
2152	0
2153	0
2154	0
2155	0

Correlation matrix

In [ ]: df[['cmte_id','cand_id']].corr()

Out[ ]:

	cmte_id	cand_id
cmte_id	1.000000	0.000000
cand_id	0.000000	1.000000

In [ ]: df[['cmte_id','cand_id']].cov()

Out[ ]:

	cmte_id	cand_id
cmte_id	0.000000	0.000000
cand_id	0.000000	0.000000

In [ ]: df[['cmte_id','cand_id']].corrwith(df.cand_id)

Out[ ]:

```
cmte_id    0.000000
cand_id    1.000000
dtype: float64
```

In [ ]: df[['cmte_id','cand_id']].corrwith(df.cand_id,axis=1).head()

Out[ ]:

```
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
dtype: float64
```

In [ ]: df[['cmte_id','cand_id']].corrwith(df.cand_id,axis=1).shape

Out[ ]: (536165,)

In [ ]: df[['cmte_id','cand_id']].corrwith(df.cand_id,axis=1).sample(10)

Out[ ]:

```
234     NaN
2146    NaN
2148    NaN
2149    NaN
2150    NaN
2151    NaN
2152    NaN
2153    NaN
2154    NaN
2155    NaN
dtype: float64
```

PCA

LDA cannot be performed on regression data, hence starting with PCA.

In [ ]: !pip install -U scikit-learn

Selecting 5 components

In [ ]: X = df[df.drop(['contb_receipt_amt'],1)] y = df['contb_receipt_amt'] %matplotlib inline import matplotlib as mpl import matplotlib.pyplot as plt import numpy as np import sklearn.decomposition as dec ncomp = 5 pca = dec.PCA(n_components=ncomp) X_r = pca.fit(X).transform(X) print('Components\n', pca.components_) print('Variance\n', pca.explained_variance_) print('Variance Ratio\n', pca.explained_variance_ratio_) print('Mean\n', pca.mean_) print('Covariance\n', pca.get_covariance()) print('Precision\n', pca.get_precision()) print('Noise Variance\n', pca.noise_variance_) print('Number of Components\n', pca.n_components_) print('Number of Features\n', pca.n_features_) print('Number of Samples\n', pca.n_samples_) print('Score\n', pca.score(X)) print('Score Samples\n', pca.score_samples(X)) print('Singular Values\n', pca.singular_values_) print('Transformed Data\n', X_r) plt.bar(np.arrow(ncomp), pca.explained_variance_ratio_, width=0.1, color='r') plt.xlabel('Principal Components') plt.ylabel('Variance Ratio') plt.title('Variance Ratio Plot') plt.show()

```
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/5e/82/74819fe32e9ec262fb9a0ae2ac62cb6a6f2b0525a0d4a8d8b3f3ef8d0a0f/scikit_learn-0.22.1-cp36-cp36m-manylinux2010_x86_64.whl (7.1MB)
     |100%| 7.1MB 1.1MB/s
Requirement already up-to-date: numpy>=1.11.0 in /usr/local/lib/python3.6/dist-packages (from scikit-learn) (1.18.1)
Requirement already up-to-date: scipy>=0.17.0 in /usr/local/lib/python3.6/dist-packages (from scikit-learn) (1.4.1)
Requirement already up-to-date: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn) (0.14.1)
Installing collected packages: scikit-learn
  Found existing installation: scikit-learn 0.22.0
    Uninstalling scikit-learn-0.22.0:
      Successfully uninstalled scikit-learn-0.22.0
Successfully installed scikit-learn-0.22.1
Components
 [[ 1.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.99999999 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 -0.99999999 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 -0.00000000 -0.99999999 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.99999999
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
Variance
 [2.00000000 1.00000000 1.00000000 1.00000000 1.00000000]
Variance Ratio
 [0.33333333 0.16666667 0.16666667 0.16666667 0.16666667]
Mean
 [ 0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
Covariance
 [[ 2.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 1.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 1.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 -0.00000000 1.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 -0.00000000 -0.00000000 1.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
Precision
 [[ 0.50000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 1.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 1.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 -0.00000000 1.00000000 -0.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
 [-0.00000000 -0.00000000 -0.00000000 -0.00000000 1.00000000
  -0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000
  -0.00000000 -0.00000000]
Noise Variance
 1.0000000000000004
Number of Components
 5
Number of Features
 12
Number of Samples
 536165
Score
 -12.000000000000004
Score Samples
 [-12.00000000 -12.00000000 -12.00000000 ... -12.00000000 -12.00000000
 -12.00000000]
Singular Values
 [1.41421356 1.00000000 1.00000000 1.00000000 1.00000000]
Transformed Data
 [[ 0.00000000 -0.00000000 -0.00000000 -0.00000000 -0.00000000]
 [
```

```
In [1]: fit_model, pp_model_matrix, predictors_model, response_model, predictors_crossed_model = fit_model(df, 'cost_gross', 'Pclass', 'cost_
        ~ Pclass')
        print('Regression')
        fit_model.summary()
        print('Model Matrix dim: %s' % pp_model_matrix.shape)
        print('Model Matrix Cols: %s' % pp_model_matrix.columns)
        print('Predictors(continuous): %s' % predictors_model)
        print('Response: %s' % response_model)
        print('Predictors(interactions): %s' % predictors_crossed_model)
```

```
Regression
                            OLS Regression Results
==============================================================================
Dep. Variable:              y   R-squared:                       0.000
Model:                    OLS   Adj. R-squared:                  0.000
Method:         Least Squares   F-statistic:                     0.000
Date:        Tue, 08 Mar 2016   Prob (F-statistic):              0.984
Time:                08:29:00   Log-Likelihood:                -8.6e+04
No. Observations:      891000   AIC:                         1.722e+05
Df Residuals:          890999   BIC:                         1.722e+05
Df Model:                   1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0000      0.000     -0.000      1.000      -0.000       0.000
Pclass         0.0000      0.000      0.000      0.984      -0.000       0.000
Model Matrix dim: (891000, 2)
Model Matrix Cols: Intercept, Pclass
Predictors(continuous): Pclass
Response: y
Predictors(interactions): None
```

```
In [2]: fit_model.fittedvalues
        fit_model.resid
```

```
Out[2]: 0    -0.000000
        1    -0.000000
        2    -0.000000
        3    -0.000000
        4    -0.000000
        dtype: float64
```

Feature Selection with Random Forest

```
In [3]: import sklearn.ensemble
        import numpy as np
        import sklearn.cross_validation
        import sklearn.grid_search
        import sklearn.metrics
        from sklearn.ensemble.base import BaseEnsemble
        from sklearn.ensemble.forest import ForestClassifier
        from sklearn.ensemble.forest import ForestRegressor
        import pandas, pandas.tseries.tools, pandas.tseries.offsets
        import pandas.DataFrame
        def rf(df, cols, y):
            X = df[cols].values
            y = df[y].values
            m = sklearn.ensemble.RandomForestClassifier(n_jobs=-1)
            print('Cross Validation Accuracy: %f' % sklearn.cross_validation.cross_val_score(m, X, y, n_jobs=-1, scoring='accuracy').
            m.fit(X, y)
            return m, X, y
        def imp(m, X, y):
            from collections import namedtuple
            from sklearn.metrics.cluster.bicluster.bicluster_ext import bicluster_ext, bicluster_error
            from sklearn.ensemble.partial_dependence import _identify_keep_indices, _max_value_for_col, partial_dependence, PartialDep
            print('Accuracy: %f' % m.score(X, y, sample_weight=None))
            print('Feature importances:')
            for i, v in enumerate(m.feature_importances_):
                print('%5d: %f' % (i, v))
```

```
In [4]: m, X, y = rf(df_train, ['cost_gross', 'Pclass'], 'survived')
        imp(m, X, y)
```

```
Cross Validation Accuracy: 0.000000
Accuracy: 0.000000
Feature importances:
    0: 0.000000
    1: 0.000000
```

```
In [5]: m, X, y = rf(df_train, ['cost_gross', 'Pclass'], 'survived')
        imp(m, X, y)
```

```
Cross Validation Accuracy: 0.000000
Accuracy: 0.000000
Feature importances:
    0: 0.000000
    1: 0.000000
```

```
In [6]: df_train[['cost_gross', 'Pclass', 'survived']].head(5)
```

Out[6]:

	cost_gross	Pclass	survived
0	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000

5 rows × 3 columns

Selecting by Features.

```
In [7]: df[df.cost_gross>0].head(5)
```

Out[7]:

	passenger_id	name	sex_female	sex_male	age_gross	fare_gross	cost_gross	survived	Pclass	sibsp	parch	ticket	cabin	embarked
0	1.000000	Braund, Mr. Owen Harris	0.000000	1.000000	22.000000	7.250000	0.000000	0.000000	3.000000	1.000000	0.000000	A/5 21171	NaN	S
1	2.000000	Cumings, Mrs. John Bradley (Florence Briggs Th...	1.000000	0.000000	38.000000	71.283300	0.000000	1.000000	1.000000	1.000000	0.000000	PC 17599	C85	C
2	3.000000	Heikkinen, Miss. Laina	1.000000	0.000000	26.000000	7.925000	0.000000	1.000000	3.000000	0.000000	0.000000	STON/O2. 3101282	NaN	S
3	4.000000	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1.000000	0.000000	35.000000	53.100000	0.000000	1.000000	1.000000	1.000000	0.000000	113803	C123	S
4	5.000000	Allen, Mr. William Henry	0.000000	1.000000	35.000000	8.050000	0.000000	0.000000	3.000000	0.000000	0.000000	373450	NaN	S

5 rows × 15 columns

```
In [8]: df[df.cost_gross>0][['cost_gross', 'survived']].head(5)
```

Out[8]:

	cost_gross	survived
0	0.000000	0.000000
1	0.000000	0.000000
2	0.000000	0.000000
3	0.000000	0.000000
4	0.000000	0.000000

2 rows × 2 columns

```
In [9]: df[df.cost_gross>0].survived.mean()
```

```
Out[9]: 0.000000
```

```
In [10]: df[df.cost_gross>0].survived.count()
```

```
Out[10]: 891000
```

```
In [11]: df[df.cost_gross>0].survived.value_counts()
```

```
Out[11]: 0    891000
         dtype: int64
```

```
In [12]: df[df.cost_gross>0].groupby('Pclass').survived.mean()
```

```
Out[12]: Pclass
         0    0.000000
         1    0.000000
         2    0.000000
         3    0.000000
         Name: survived, dtype: float64
```

```
In [13]: df[df.cost_gross>0].groupby('Pclass').survived.count()
```

```
Out[13]: Pclass
         0    891000
         1         0
         2         0
         3         0
         Name: survived, dtype: int64
```

```
In [14]: df[df.cost_gross>0].groupby('Pclass').survived.value_counts()
```

```
Out[14]: Pclass  survived
         0       0.000000    891000
         1       0.000000         0
         2       0.000000         0
         3       0.000000         0
         Name: survived, dtype: int64
```

```
In [15]: df[df.cost_gross>0].groupby('Pclass').survived.value_counts().unstack
```

Out[15]:

survived	0.000000	0.000000	0.000000	0.000000
Pclass	0	1	2	3
0.000000	891000	NaN	NaN	NaN

```
In [16]: df[df.cost_gross>0].groupby('Pclass').survived.value_counts().unstack.plot(kind='bar')
```

```
Out[16]: 
```

```
In [47]: df[5:]
```

	stop_short_circ	coup_fly	stop_fly	stop_short_rotor	grass_thrusts	coup_thrusts	TotalCombatSuccess	stop_TotalCombatSuccess	stop_...
5	0.12121	0.78788	0.52273	0.73636	0.57576	0.53030	0.65455	0.56250	0
6	0.12121	0.78788	0.52273	0.73636	0.57576	0.53030	0.65455	0.56250	0
7	0.12121	0.78788	0.52273	0.73636	0.57576	0.53030	0.65455	0.56250	0
8	0.12121	0.78788	0.52273	0.73636	0.57576	0.53030	0.65455	0.56250	0
9	0.12121	0.78788	0.52273	0.73636	0.57576	0.53030	0.65455	0.56250	0
...	...	...	...	...	...	...	...	...	...
2130	0.07625	0.69630	0.61342	0.76629	0.76629	0.76629	0.76629	0.76629	0
2131	0.07625	0.69630	0.61342	0.76629	0.76629	0.76629	0.76629	0.76629	0
2132	0.07625	0.69630	0.61342	0.76629	0.76629	0.76629	0.76629	0.76629	0
2133	0.07625	0.69630	0.61342	0.76629	0.76629	0.76629	0.76629	0.76629	0
2134	0.07625	0.69630	0.61342	0.76629	0.76629	0.76629	0.76629	0.76629	0

2130 rows × 20 columns

```
In [48]: pd.set_option("display.max_columns",20)
```

```
In [49]: pd.set_option("display.precision",4)
```

```
In [50]: from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor, AdaBoostRegressor, GradientBoostingRegressor
```

```
In [51]: ensemble = {'Random Forest':RandomForestRegressor(),
         'Extra Trees':ExtraTreesRegressor(),
         'AdaBoost':AdaBoostRegressor(),
         'Gradient Boosting':GradientBoostingRegressor()}
```

```
In [52]: for name,model in ensemble.items:
             print(name,model.fit(X,Y).score(X,Y))
```

```
Random Forest 0.9126
Extra Trees 1.0000
AdaBoost 0.7127
Gradient Boosting 0.9126
```

Selecting K-Features

```
In [53]: from sklearn.feature_selection import SelectKBest,f_regression
```

```
In [54]: for k in range(1,21):
             print("\n",k,"-Features\n",sorted(SelectKBest(f_regression,k).fit(X,Y).get_support(True),key=lambda x:X.columns[x]),sep="")
```

```
 1-Features
 ['stop_TotalCombatSuccess']

 2-Features
 ['stop_short_circ', 'stop_TotalCombatSuccess']

 3-Features
 ['stop_short_circ', 'stop_fly', 'stop_TotalCombatSuccess']

 4-Features
 ['stop_short_circ', 'stop_fly', 'stop_rotor', 'stop_TotalCombatSuccess']

 5-Features
 ['stop_short_circ', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'stop_TotalCombatSuccess']

 6-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'stop_TotalCombatSuccess']

 7-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'stop_TotalCombatSuccess']

 8-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess']

 9-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts']

 10-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi']

 11-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki']

 12-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio']

 13-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio']

 14-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio']

 15-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio']

 16-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio']

 17-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio']

 18-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio']

 19-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio']

 20-Features
 ['stop_short_circ', 'coup_fly', 'stop_fly', 'stop_rotor', 'grass_thrusts', 'coup_thrusts', 'TotalCombatSuccess', 'stop_TotalCombatSuccess', 'stop_thrusts', 'stop_Kamaitachi', 'stop_Kamaitachi_Ki', 'stop_Kamaitachi_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio', 'stop_Kamaitachi_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio_Ki_Ratio']
```

```
In [55]: df[sorted(SelectKBest(f_regression,k).fit(X,Y).get_support(True),key=lambda x:X.columns[x])].corrwith(Y).sort_values(ascending=False)
```

```
In [56]: from sklearn.preprocessing import StandardScaler
```

```
In [57]: Z = StandardScaler().fit(X).transform(X)
```

```
In [58]: from sklearn.linear_model import LinearRegression
```

```
In [59]: for k in range(1,21):
             print("\n",k,"-Features\n",LinearRegression().fit(Z[:,sorted(SelectKBest(f_regression,k).fit(X,Y).get_support(True)),],Y).score(Z[:,sorted(SelectKBest(f_regression,k).fit(X,Y).get_support(True)),],Y),sep="")
```

```
 1-Features
 0.5455

 2-Features
 0.6164

 3-Features
 0.6164

 4-Features
 0.6164

 5-Features
 0.6164

 6-Features
 0.6164

 7-Features
 0.6164

 8-Features
 0.6164

 9-Features
 0.6164

 10-Features
 0.6164

 11-Features
 0.6164

 12-Features
 0.6164

 13-Features
 0.6164

 14-Features
 0.6164

 15-Features
 0.6164

 16-Features
 0.6164

 17-Features
 0.6164

 18-Features
 0.6164

 19-Features
 0.6164

 20-Features
 0.6164
```

```
In [60]: %matplotlib
```

```
Using matplotlib backend: Qt5Agg
```

```
In [61]: pd.Series(LinearRegression().fit(Z[:,sorted(SelectKBest(f_regression,k).fit(X,Y).get_support(True),],Y).coef_,index=df[sorted(SelectKBest(f_regression,k).fit(X,Y).get_support(True),key=lambda x:X.columns[x])].columns).sort_values(ascending=False).plot(kind="barh"),
         plt.title("Linear Regression Coefficients\n Performance of K-Features Feature selection\n Dataset 3 Dataset of Dragon Ball Z Fighters\n Dataset 3 Dataset of Dragon Ball Z Fighters\n Dataset 3 Dataset of Dragon Ball Z Fighters
```

```
# ipd: plot_pairwise_scatter_matrix(df=df, cols=c("Prog_ipd_Pos_Prime_cls_type", "Tissue", "Accession")[1:3])
# [1] "Prog_ipd_Pos_Prime_cls_type"
# [2] "Tissue"
# [3] "Accession"
```

```
In [24]: df_cat_num_dist(df_train[['Sex']],df_train[['Survived']],df_train[['Pclass']],df_train[['Embarked']],df_train[['SibSp']],df_train[['Parch']],df_train[['Ticket']],df_train[['Cabin']],df_train[['Name']])
```

```
Sex
Survived  0  1
Sex
female    1  0
male      0  1
Pclass
Survived  0  1
Pclass
1         1  0
2         1  0
3         0  1
Embarked
Survived  0  1
Embarked
C         1  0
Q         1  0
S         0  1
SibSp
Survived  0  1
SibSp
0         1  0
1         1  0
2         1  0
3         1  0
4         1  0
5         1  0
8         1  0
Parch
Survived  0  1
Parch
0         1  0
1         1  0
2         1  0
3         1  0
4         1  0
5         1  0
6         1  0
Ticket
Survived  0  1
Ticket
110413    1  0
113503    1  0
113781    1  0
19950     1  0
347082    1  0
350406    1  0
382652    1  0
C.A. 2673 1  0
C.A. 33112 1  0
CA 2144   1  0
CA. 2343  1  0
PC 17757  1  0
PC 17760  1  0
PC 17761  1  0
PC 17582  1  0
PC 17611  1  0
SC/Paris 2123 1  0
SOTON/O.Q. 3101305 1  0
STON/O2. 3101282 1  0
W./C. 6608 1  0
W./C. 6609 1  0
Cabin
Survived  0  1
Cabin
B96 B98   1  0
C23 C25 C27 1  0
E101      1  0
F33       1  0
G6        1  0
Name
Survived  0  1
Name
Abbott, Master. Eugene Joseph 1  0
Abbott, Mr. Rossmore Edward 1  0
Abbott, Mrs. Stanton (Rosa Hunt Wilson) 1  0
Abelseth, Miss. Karen Marie 1  0
Abelseth, Mr. Olaf 1  0
Aks, Mrs. Sam (Leah Rosen) 1  0
Allen, Miss. Elisabeth Walton 1  0
Allen, Mr. William Henry 1  0
Andrews, Miss. Kornelia Theodosia 1  0
Andrews, Mr. Thomas jr 1  0
Angle, Mrs. William A (Florence "Mary" Agnes Hughes) 1  0
Baclini, Miss. Eugenie 1  0
Baclini, Miss. Marie Catherine 1  0
Baclini, Mrs. Elias (Marie Grandica) 1  0
Bailey, Mr. Percy Andrew 1  0
Baker, Master. Alexander R. 1  0
Baker, Mrs. Bertram (Suzanne Adelaide Duval) 1  0
Barber, Miss. Ellen "Nellie" 1  0
Barkworth, Mr. Algernon Henry Wilson 1  0
Baxter, Mr. Quigg Edmond 1  0
Beane, Mr. Edward 1  0
Becker, Master. Richard F 1  0
Becker, Miss. Marion Louise 1  0
Becker, Mrs. Allen (Marion Grace Thompson) 1  0
Beckwith, Mr. Richard Leonard 1  0
Bentham, Miss. Lilian W 1  0
Berg, Miss. Marta Katarina 1  0
Berg, Mr. Karl Ola 1  0
Bernstein, Mr. Adolf 1  0
Bishop, Mr. Dickinson H 1  0
Bishop, Mrs. Dickinson H (Helen Walton) 1  0
Bjorklund, Mr. Ernst Gustaf Adolf 1  0
Bjornstrom-Steffansson, Mr. Karl Johan 1  0
Blackwell, Mr. Stephen Weart 1  0
Blank, Mr. Henry 1  0
Bonnell, Miss. Caroline 1  0
Bonnell, Miss. Elizabeth 1  0
Bowerman, Miss. Elsie Edith 1  0
Bradley, Mr. George ("George Arthur Brayton") 1  0
Brown, Miss. Amelia "Mildred" 1  0
Brown, Miss. Edith Eileen 1  0
Brown, Mrs. James Joseph (Margaret Tobin) 1  0
Brown, Mrs. Thomas William Solomon (Elizabeth Catherine Ford Gilinski) 1  0
Butt, Major. Archibald Willingham 1  0
Butt, Mr. Laurence Mark 1  0
Calderhead, Mr. Edward Pennington 1  0
Carter, Master. William Thornton II 1  0
Carter, Mrs. William Ernest (Lucile Polk) 1  0
Carter, Rev. Ernest Courtenay 1  0
Cavendish, Mrs. Tyrell William (Julia Florence Siegel) 1  0
Chaffee, Mr. Herbert Fuller 1  0
Chaffee, Mrs. Herbert Fuller (Carrie Constance Toogood) 1  0
Chambers, Mrs. Norman Campbell (Bessie Bowen) 1  0
Chibnall, Mrs. (Edith Martha Bowerman) 1  0
Chisholm, Mrs. Ernest Harold (Violet Beatrix Kinnan) 1  0
Clark, Mr. Walter Miller 1  0
Clarke, Mrs. Charles V (Ada Maria Winfield) 1  0
Cleaver, Miss. Alice 1  0
Clifford, Mr. George Quincy 1  0
Cornell, Mrs. Robert Clifford (Malvina Helena Vanschoiick) 1  0
Crosby, Capt. Edward Gifford 1  0
Crosby, Miss. Harriet R. 1  0
Cumings, Mrs. John Bradley (Florence Briggs Thayer) 1  0
Dick, Mr. Albert Adrian 1  0
Dick, Mrs. Albert Adrian (Vera Gillespie) 1  0
Dodge, Master. Washington 1  0
Dodge, Mrs. Washington (Ruth Vidaver) 1  0
Douglas, Mr. Walter Donald 1  0
Douglas, Mrs. Frank Milley (Mary Agnes Peacock) 1  0
Dyker, Mrs. Adolf Fredrik (Anna Elisabeth Judith Andersdotter) 1  0
Earnshaw, Mrs. Boulton (Olive Potter Cathcart) 1  0
Eustis, Miss. Elizabeth Mussey 1  0
Fleming, Miss. Margaret 1  0
Fortune, Mrs. Mark (Mary McDougald) 1  0
Frolicher-Stehli, Mr. Maxmilian 1  0
Frolicher-Stehli, Mrs. Maxmilian (Margaretha Emerentia Stehli) 1  0
Gale, Mr. Harry 1  0
Gibson, Miss. Dorothy 1  0
Gibson, Mrs. Leonard (Pauline C Boeson) 1  0
Giglio, Mr. Victor 1  0
Greenfield, Mrs. Leo David (Malkah Rattner) 1  0
Harder, Mr. George Achilles 1  0
Harder, Mrs. George Achilles (Dorothy Anna Wilkinson) 1  0
Harper, Mrs. Henry Sleeper (Myra Mary Fox) 1  0
Harris, Mrs. Henry Birkhardt (Irene Harris-Rappaport) 1  0
Hart, Mrs. Benjamin (Esther Ada Bradley) 1  0
Hippach, Miss. Jean Gertrude 1  0
Hippach, Mrs. Louis Guillaume (Clara Jennings Gerhardt) 1  0
Hoyt, Mrs. Frederick Maxfield (Jane Anne Forby-Ramsey) 1  0
Ilett, Miss. Bertha 1  0
Ilett, Mr. Bertram 1  0
Ivanoff, Mr. Kanio 1  0
Jacobsohn, Mrs. Sidney Samuel (Amy Frankenthaler) 1  0
Jarvis, Mr. John Denzil 1  0
Jefferys, Mr. Clifford Thomas 1  0
Jefferys, Mr. Ernest Wilfrid 1  0
Jussila, Miss. Mari Aina 1  0
Kantor, Mrs. Sinai (Miriam Sternin) 1  0
Kelly, Mr. James 1  0
Kelly, Mr. James Bernard 1  0
Kelly, Mrs. Florence "Fannie" 1  0
Kenyon, Mrs. Frederick R (Marion) 1  0
Kimball, Mrs. Edwin Nelson Jr (Gertrude Parsons) 1  0
Klaber, Mr. Herman 1  0
Klaber, Mrs. Herman (Caroline Lesser) 1  0
Kreuchen, Miss. Emilie 1  0
Lahtinen, Mrs. William (Anna Sofia Maria Panula) 1  0
Lehmann, Miss. Bertha 1  0
Lehmann, Mr. Rudolf 1  0
Lehmann, Mrs. John George (Bertha E H Mulder) 1  0
Lemberopolous, Mr. Peter L 1  0
Lesurer, Mr. Gustave J 1  0
Lewy, Mr. Ervin G. 1  0
Lindblom, Miss. Augusta Charlotta 1  0
Longley, Miss. Gretchen Fiske 1  0
Lurette, Miss. Elise 1  0
McCarthy, Miss. Catherine Katie 1  0
McCoy, Miss. Alicia 1  0
McCoy, Mr. Bernard 1  0
McCoy, Mr. Harry 1  0
McDermott, Miss. Brigdet Delia 1  0
McGough, Mr. James Robert 1  0
McKane, Miss. Adelaide 1  0
Minahan, Dr. William Edward 1  0
Minahan, Miss. Daisy E 1  0
Mock, Mr. Philipp Edmund 1  0
Moen, Mr. Sigurd Hansen 1  0
Mortdecai, Mr. Emil 1  0
Mulvihill, Miss. Bridget Doyle 1  0
Nasser, Mr. Nicholas 1  0
Navratil, Master. Michel Marcel 1  0
Navratil, Mr. Michel ("Louis M Hoffman") 1  0
Navratil, Mr. Michel (Tom London") 1  0
Newell, Miss. Madeleine 1  0
Newell, Miss. Marjorie 1  0
Nicola-Yarred, Master. Elias 1  0
Nicola-Yarred, Miss. Jamila 1  0
Nilsson, Miss. Berta Olivia 1  0
Nourney, Mr. Alfred (Baron von Drachstedt") 1  0
Nye, Mrs. (Elizabeth Ramell) 1  0
Oliva y Ocana, Dona. Fermina 1  0
Ostby, Miss. Helene R 1  0
Ostby, Mr. Engelhart Cornelius 1  0
Pain, Dr. Alfred 1  0
Pain, Mrs. Alfred (Emma Louise Lamy) 1  0
Pavlovic, Mr. Stefan 1  0
Pavlovic, Mrs. Stefan (Anna Kristine) 1  0
Pengelly, Mr. Frederick William 1  0
Penn, Mrs. William (Hannah Stockton Phillips) 1  0
Perkin, Mr. John Henry 1  0
Phillips, Miss. Kate Florence ("Mrs Kate Louise Phillips Marshall") 1  0
Pinsky, Mrs. (Rosa) 1  0
Porter, Mr. Walter Chamberlain 1  0
Quick, Miss. Winifred Vera 1  0
Quick, Mrs. Frederick Charles (Jane Richards) 1  0
Ramell, Mr. Anders Johan 1  0
Rice, Master. Eugene 1  0
Rice, Master. George Hugh 1  0
Rice, Master. Arthur 1  0
Rice, Mrs. William (Margaret Norton) 1  0
Rommetvedt, Mr. Knud Paustian 1  0
Rosblom, Mrs. Viktor (Helena Wilhelmina) 1  0
Rosblom, Miss. Salli Helena 1  0
Rosblom, Mr. Viktor Richard 1  0
Ryerson, Miss. Emily Borie 1  0
Ryerson, Miss. Susanna Haven 1  0
Ryerson, Mr. Arthur Larned 1  0
Ryerson, Mrs. Arthur Larned (Emma Jane Buckland) 1  0
Saalfeld, Mr. Adolphe 1  0
Sagesser, Mlle. Emma 1  0
Sandstrom, Miss. Marguerite Rut 1  0
Sawyer, Mrs. (Hulda Margaretha Garborg) 1  0
Schabert, Mrs. Paul (Emma Sofie Schroeder) 1  0
Shelley, Mrs. William (Imanita Parrish Hall) 1  0
Silven, Miss. Lyyli Karoliina 1  0
Silvey, Mr. William Baker 1  0
Silvey, Mrs. William Baker (Alice Murphy Dunnigan) 1  0
Simonius-Blumer, Col. Oberst Alfred 1  0
Slayter, Miss. Hilda Mary 1  0
Slemen, Mr. Richard James 1  0
Smart, Mrs. John (Elizabeth Lamb Imanita Parrish Hall) 1  0
Smith, Mr. James Clemens 1  0
Solomon, Mrs. Solomon (Rebecca Napier Lyon) 1  0
Stahelin-Maeglin, Dr. Max 1  0
Stead, Mr. William Thomas 1  0
Stephenson, Mrs. Walter Bertram (Martha Eustis) 1  0
Stewart, Mr. Albert A 1  0
Sutehall, Mr. Henry Jr 1  0
Taussig, Mrs. Emil (Tillie Mandelbaum) 1  0
Taussig, Mrs. Emil (Tillie Mandelbaum) 1  0
Taussig, Mrs. Emil (Tillie Mandelbaum) 1  0
Thayer, Mr. John Borland Jr 1  0
Thayer, Mrs. John Borland (Marian Longstreth Morris) 1  0
Thorne, Mrs. Gertrude Maybelle 1  0
Tornquist, Mr. William Henry 1  0
Troupiansky, Mr. Moses Aaron 1  0
Troupiansky, Mrs. Moses Aaron (Esther) 1  0
Trout, Mrs. William H (Jessie L) 1  0
Turpin, Mr. William John Robert 1  0
Turpin, Mrs. William John Robert (Dorothy Ann Wonnacott) 1  0
Uruchurtu, Don. Manuel E 1  0
Van Billiard, Master. James William 1  0
Van Billiard, Mr. Austin Blyler 1  0
Warren, Mrs. Frank Manley (Anna Sophia Atkinson) 1  0
Webber, Miss. Susan 1  0
Wheeler, Mr. Edwin Frank 1  0
Wick, Mrs. George Dennick (Mary Hitchcock) 1  0
Wick, Miss. Mary Natalie 1  0
Wilkes, Mrs. James (Ellen Needs) 1  0
Williams, Mr. Leslie 1  0
Wright, Miss. Marguerite Irene 1  0
Yasbeck, Mrs. Antoni (Selini Alexander) 1  0
Young, Miss. Marie Grice 1  0
Zabour, Miss. Hileni 1  0
Zabour, Miss. Theresia 1  0
Zimmerman, Mr. Leo 1  0
```

```
In [25]: df_cat_num_dist(df_test[['Sex']],df_test[['Pclass']],df_test[['Embarked']],df_test[['SibSp']],df_test[['Parch']],df_test[['Ticket']],df_test[['Cabin']],df_test[['Name']])
```

```
Sex
Survived  0  1
Sex
female    1  0
male      0  1
Pclass
Survived  0  1
Pclass
1         1  0
2         1  0
3         0  1
Embarked
Survived  0  1
Embarked
C         1  0
Q         1  0
S         0  1
SibSp
Survived  0  1
SibSp
0         1  0
1         1  0
2         1  0
3         1  0
4         1  0
5         1  0
8         1  0
Parch
Survived  0  1
Parch
0         1  0
1         1  0
2         1  0
3         1  0
4         1  0
5         1  0
6         1  0
Ticket
Survived  0  1
Ticket
110413    1  0
113503    1  0
113781    1  0
19950     1  0
347082    1  0
350406    1  0
382652    1  0
C.A. 2673 1  0
C.A. 33112 1  0
CA 2144   1  0
CA. 2343  1  0
PC 17757  1  0
PC 17760  1  0
PC 17761  1  0
PC 17582  1  0
PC 17611  1  0
SC/Paris 2123 1  0
SOTON/O.Q. 3101305 1  0
STON/O2. 3101282 1  0
W./C. 6608 1  0
W./C. 6609 1  0
Cabin
Survived  0  1
Cabin
B96 B98   1  0
C23 C25 C27 1  0
E101      1  0
F33       1  0
G6        1  0
Name
Survived  0  1
Name
Abbott, Master. Eugene Joseph 1  0
Abbott, Mr. Rossmore Edward 1  0
Abbott, Mrs. Stanton (Rosa Hunt Wilson) 1  0
Abelseth, Miss. Karen Marie 1  0
Abelseth, Mr. Olaf 1  0
Aks, Mrs. Sam (Leah Rosen) 1  0
Allen, Miss. Elisabeth Walton 1  0
Allen, Mr. William Henry 1  0
Andrews, Miss. Kornelia Theodosia 1  0
Andrews, Mr. Thomas jr 1  0
Angle, Mrs. William A (Florence "Mary" Agnes Hughes) 1  0
Baclini, Miss. Eugenie 1  0
Baclini, Miss. Marie Catherine 1  0
Baclini, Mrs. Elias (Marie Grandica) 1  0
Bailey, Mr. Percy Andrew 1  0
Baker, Master. Alexander R. 1  0
Baker, Mrs. Bertram (Suzanne Adelaide Duval) 1  0
Barber, Miss. Ellen "Nellie" 1  0
Barkworth, Mr. Algernon Henry Wilson 1  0
Baxter, Mr. Quigg Edmond 1  0
Beane, Mr. Edward 1  0
Becker, Master. Richard F 1  0
Becker, Miss. Marion Louise 1  0
Becker, Mrs. Allen (Marion Grace Thompson) 1  0
Beckwith, Mr. Richard Leonard 1  0
Bentham, Miss. Lilian W 1  0
Berg, Miss. Marta Katarina 1  0
Berg, Mr. Karl Ola 1  0
Bernstein, Mr. Adolf 1  0
Bishop, Mr. Dickinson H 1  0
Bishop, Mrs. Dickinson H (Helen Walton) 1  0
Bjorklund, Mr. Ernst Gustaf Adolf 1  0
Bjornstrom-Steffansson, Mr. Karl Johan 1  0
Blackwell, Mr. Stephen Weart 1  0
Blank, Mr. Henry 1  0
Bonnell, Miss. Caroline 1  0
Bonnell, Miss. Elizabeth 1  0
Bowerman, Miss. Elsie Edith 1  0
Bradley, Mr. George ("George Arthur Brayton") 1  0
Brown, Miss. Amelia "Mildred" 1  0
Brown, Miss. Edith Eileen 1  0
Brown, Mrs. James Joseph (Margaret Tobin) 1  0
Brown, Mrs. Thomas William Solomon (Elizabeth Catherine Ford Gilinski) 1  0
Butt, Major. Archibald Willingham 1  0
Butt, Mr. Laurence Mark 1  0
Calderhead, Mr. Edward Pennington 1  0
Carter, Master. William Thornton II 1  0
Carter, Mrs. William Ernest (Lucile Polk) 1  0
Carter, Rev. Ernest Courtenay 1  0
Cavendish, Mrs. Tyrell William (Julia Florence Siegel) 1  0
Chaffee, Mr. Herbert Fuller 1  0
Chaffee, Mrs. Herbert Fuller (Carrie Constance Toogood) 1  0
Chambers, Mrs. Norman Campbell (Bessie Bowen) 1  0
Chibnall, Mrs. (Edith Martha Bowerman) 1  0
Chisholm, Mrs. Ernest Harold (Violet Beatrix Kinnan) 1  0
Clark, Mr. Walter Miller 1  0
Clarke, Mrs. Charles V (Ada Maria Winfield) 1  0
Cleaver, Miss. Alice 1  0
Clifford, Mr. George Quincy 1  0
Cornell, Mrs. Robert Clifford (Malvina Helena Vanschoiick) 1  0
Crosby, Capt. Edward Gifford 1  0
Crosby, Miss. Harriet R. 1  0
Cumings, Mrs. John Bradley (Florence Briggs Thayer) 1  0
Dick, Mr. Albert Adrian 1  0
Dick, Mrs. Albert Adrian (Vera Gillespie) 1  0
Dodge, Master. Washington 1  0
Dodge, Mrs. Washington (Ruth Vidaver) 1  0
Douglas, Mr. Walter Donald 1  0
Douglas, Mrs. Frank Milley (Mary Agnes Peacock) 1  0
Dyker, Mrs. Adolf Fredrik (Anna Elisabeth Judith Andersdotter) 1  0
Earnshaw, Mrs. Boulton (Olive Potter Cathcart) 1  0
Eustis, Miss. Elizabeth Mussey 1  0
Fleming, Miss. Margaret 1  0
Fortune, Mrs. Mark (Mary McDougald) 1  0
Frolicher-Stehli, Mr. Maxmilian 1  0
Frolicher-Stehli, Mrs. Maxmilian (Margaretha Emerentia Stehli) 1  0
Gale, Mr. Harry 1  0
Gibson, Miss. Dorothy 1  0
Gibson, Mrs. Leonard (Pauline C Boeson) 1  0
Giglio, Mr. Victor 1  0
Greenfield, Mrs. Leo David (Malkah Rattner) 1  0
Harder, Mr. George Achilles 1  0
Harder, Mrs. George Achilles (Dorothy Anna Wilkinson) 1  0
Harper, Mrs. Henry Sleeper (Myra Mary Fox) 1  0
Harris, Mrs. Henry Birkhardt (Irene Harris-Rappaport) 1  0
Hart, Mrs. Benjamin (Esther Ada Bradley) 1  0
Hippach, Miss. Jean Gertrude 1  0
Hippach, Mrs. Louis Guillaume (Clara Jennings Gerhardt) 1  0
Hoyt, Mrs. Frederick Maxfield (Jane Anne Forby-Ramsey) 1  0
Ilett, Miss. Bertha 1  0
Ilett, Mr. Bertram 1  0
Ivanoff, Mr. Kanio 1  0
Jacobsohn, Mrs. Sidney Samuel (Amy Frankenthaler) 1  0
Jarvis, Mr. John Denzil 1  0
Jefferys, Mr. Clifford Thomas 1  0
Jefferys, Mr. Ernest Wilfrid 1  0
Jussila, Miss. Mari Aina 1  0
Kantor, Mrs. Sinai (Miriam Sternin) 1  0
Kelly, Mr. James 1  0
Kelly, Mr. James Bernard 1  0
Kelly, Mrs. Florence "Fannie" 1  0
Kenyon, Mrs. Frederick R (Marion) 1  0
Kimball, Mrs. Edwin Nelson Jr (Gertrude Parsons) 1  0
Klaber, Mr. Herman 1  0
Klaber, Mrs. Herman (Caroline Lesser) 1  0
Kreuchen, Miss. Emilie 1  0
Lahtinen, Mrs. William (Anna Sofia Maria Panula) 1  0
Lehmann, Miss. Bertha 1  0
Lehmann, Mr. Rudolf 1  0
Lehmann, Mrs. John George (Bertha E H Mulder) 1  0
Lemberopolous, Mr. Peter L 1  0
Lesurer, Mr. Gustave J 1  0
Lewy, Mr. Ervin G. 1  0
Lindblom, Miss. Augusta Charlotta 1  0
Longley, Miss. Gretchen Fiske 1  0
Lurette, Miss. Elise 1  0
McCarthy, Miss. Catherine Katie 1  0
McCoy, Miss. Alicia 1  0
McCoy, Mr. Bernard 1  0
McCoy, Mr. Harry 1  0
McDermott, Miss. Brigdet Delia 1  0
McGough, Mr. James Robert 1  0
McKane, Miss. Adelaide 1  0
Minahan, Dr. William Edward 1  0
Minahan, Miss. Daisy E 1  0
Mock, Mr. Philipp Edmund 1  0
Moen, Mr. Sigurd Hansen 1  0
Mortdecai, Mr. Emil 1  0
Mulvihill, Miss. Bridget Doyle 1  0
Nasser, Mr. Nicholas 1  0
Navratil, Master. Michel Marcel 1  0
Navratil, Mr. Michel ("Louis M Hoffman") 1  0
Navratil, Mr. Michel (Tom London") 1  0
Newell, Miss. Madeleine 1  0
Newell, Miss. Marjorie 1  0
Nicola-Yarred, Master. Elias 1  0
Nicola-Yarred, Miss. Jamila 1  0
Nilsson, Miss. Berta Olivia 1  0
Nourney, Mr. Alfred (Baron von Drachstedt") 1  0
Nye, Mrs. (Elizabeth Ramell) 1  0
Oliva y Ocana, Dona. Fermina 1  0
Ostby, Miss. Helene R 1  0
Ostby, Mr. Engelhart Cornelius 1  0
Pain, Dr. Alfred 1  0
Pain, Mrs. Alfred (Emma Louise Lamy) 1  0
Pavlovic, Mr. Stefan 1  0
Pavlovic, Mrs. Stefan (Anna Kristine) 1  0
Pengelly, Mr. Frederick William 1  0
Penn, Mrs. William (Hannah Stockton Phillips) 1  0
Perkin, Mr. John Henry 1  0
Phillips, Miss. Kate Florence ("Mrs Kate Louise Phillips Marshall") 1  0
Pinsky, Mrs. (Rosa) 1  0
Porter, Mr. Walter Chamberlain 1  0
Quick, Miss. Winifred Vera 1  0
Quick, Mrs. Frederick Charles (Jane Richards) 1  0
Ramell, Mr. Anders Johan 1  0
Rice, Master. Eugene 1  0
Rice, Master. George Hugh 1  0
Rice, Master. Arthur 1  0
Rice, Mrs. William (Margaret Norton) 1  0
Rommetvedt, Mr. Knud Paustian 1  0
Rosblom, Mrs. Viktor (Helena Wilhelmina) 1  0
Rosblom, Miss. Salli Helena 1  0
Rosblom, Mr. Viktor Richard 1  0
Ryerson, Miss. Emily Borie 1  0
Ryerson, Miss. Susanna Haven 1  0
Ryerson, Mr. Arthur Larned 1  0
Ryerson, Mrs. Arthur Larned (Emma Jane Buckland) 1  0
Saalfeld, Mr. Adolphe 1  0
Sagesser, Mlle. Emma 1  0
Sandstrom, Miss. Marguerite Rut 1  0
Sawyer, Mrs. (Hulda Margaretha Garborg) 1  0
Schabert, Mrs. Paul (Emma Sofie Schroeder) 1  0
Shelley, Mrs. William (Imanita Parrish Hall) 1  0
Silven, Miss. Lyyli Karoliina 1  0
Silvey, Mr. William Baker 1  0
Silvey, Mrs. William Baker (Alice Murphy Dunnigan) 1  0
Simonius-Blumer, Col. Oberst Alfred 1  0
Slayter, Miss. Hilda Mary 1  0
Slemen, Mr. Richard James 1  0
Smart, Mrs. John (Elizabeth Lamb Imanita Parrish Hall) 1  0
Smith, Mr. James Clemens 1  0
Solomon, Mrs. Solomon (Rebecca Napier Lyon) 1  0
Stahelin-Maeglin, Dr. Max 1  0
Stead, Mr. William Thomas 1  0
Stephenson, Mrs. Walter Bertram (Martha Eustis) 1  0
Stewart, Mr. Albert A 1  0
Sutehall, Mr. Henry Jr 1  0
Taussig, Mrs. Emil (Tillie Mandelbaum) 1  0
Taussig, Mrs. Emil (Tillie Mandelbaum) 1  0
Taussig, Mrs. Emil (Tillie Mandelbaum) 1  0
Thayer, Mr. John Borland Jr 1  0
Thayer, Mrs. John Borland (Marian Longstreth Morris) 1  0
Thorne, Mrs. Gertrude Maybelle 1  0
Tornquist, Mr. William Henry 1  0
Troupiansky, Mr. Moses Aaron 1  0
Troupiansky, Mrs. Moses Aaron (Esther) 1  0
Trout, Mrs. William H (Jessie L) 1  0
Turpin, Mr. William John Robert 1  0
Turpin, Mrs. William John Robert (Dorothy Ann Wonnacott) 1  0
Uruchurtu, Don. Manuel E 1  0
Van Billiard, Master. James William 1  0
Van Billiard, Mr. Austin Blyler 1  0
Warren, Mrs. Frank Manley (Anna Sophia Atkinson) 1  0
Webber, Miss. Susan 1  0
Wheeler, Mr. Edwin Frank 1  0
Wick, Mrs. George Dennick (Mary Hitchcock) 1  0
Wick, Miss. Mary Natalie 1  0
Wilkes, Mrs. James (Ellen Needs) 1  0
Williams, Mr. Leslie 1  0
Wright, Miss. Marguerite Irene 1  0
Yasbeck, Mrs. Antoni (Selini Alexander) 1  0
Young, Miss. Marie Grice 1  0
Zabour, Miss. Hileni 1  0
Zabour, Miss. Theresia 1  0
Zimmerman, Mr. Leo 1  0
```

LINE

```
In [26]: df_num_dist(df_train[['Age']],df_train[['Fare']])
```

```
In [27]: df_num_dist(df_test[['Age']],df_test[['Fare']])
```

Performing cross-validation for 5 folds and 10 components respectively.

```
In [28]: from sklearn.preprocessing import StandardScaler
         from sklearn.decomposition import PCA
         from sklearn.model_selection import cross_validation
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import ExtraTreesClassifier

         # Extract all features which have no null values
         cols = ['Age',
                 'Fare',
                 'Pclass',
                 'SibSp',
                 'Parch',
                 'Sex',
                 'Embarked']

         def train_and_evaluate(df, n_estimators, n_folds=5):
             print("Training and evaluating ...")
             print("Number of estimators: %d" % n_estimators)
             print("Number of folds: %d" % n_folds)
             global cols
             X = df[cols].fillna(-1000).values
             y = df['Survived'].values
             print("Dimensions of feature matrix: %dx%d" % X.shape)
             print("Dimensions of response vector: %d" % len(y))
             clf = RandomForestClassifier(n_estimators=n_estimators, n_jobs=-1)
             print("Classifier: %s" % clf)
             scaler = StandardScaler()
             X = scaler.fit_transform(X)
             print("Feature vectors standardized.")
             pca = PCA(n_components=10)
             X = pca.fit_transform(X)
             print("Dimensions after PCA: %dx%d" % X.shape)
             print("Explained variance ratio: %s" % pca.explained_variance_ratio_)
             print("Cross validation ...")
             scores = cross_validation.cross_val_score(clf, X, y, cv=n_folds, n_jobs=-1)
             print("Scores: %s" % scores)
             print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
             return clf, X, y, scaler, pca

         clf, X, y, scaler, pca = train_and_evaluate(df_train, n_estimators=1000)
         print("Feature importances ...")
         for i, f in enumerate(clf.feature_importances_):
             print("%d. %s %f" % (i + 1, cols[i], f))
         print("Finished.")
         print("Dimensions of test set before preprocessing: %dx%d" % df_test.shape)
         X = df_test[cols].fillna(-1000).values
         X = scaler.transform(X)
         print("Dimensions after standardization: %dx%d" % X.shape)
         X = pca.transform(X)
         print("Dimensions after PCA: %dx%d" % X.shape)
         y = clf.predict(X)
         print("Predictions ...")
         print(y)
         print("Finished.")
         df_test['Survived'] = y
         print("Saving predictions ...")
         df_test[['PassengerId', 'Survived']].to_csv('./data/titanic_pca.csv', index=False)
         print("Finished.")
```

```
Feature importances ...
1. Age 0.000000
2. Fare 0.000000
3. Pclass 0.000000
4. SibSp 0.000000
5. Parch 0.000000
6. Sex 0.000000
7. Embarked 0.000000
Finished.
Dimensions of test set before preprocessing: 418x11
Dimensions after standardization: 418x7
Dimensions after PCA: 418x10
Predictions ...
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 
```

```
[1]: fit_model(rf_tune,lr_tune,knn_tune,dtree_tune,svm_tune,
     metrics=c("accuracy","f1","recall","precision","roc_auc"),
     resamples=resamples,
     control=control_resamples(save_pred=TRUE,verbose=TRUE,extract=function(x)extract_fit_parsnip(x)$fit),
     verbose=TRUE)
```

```
Error in fit_model(rf_tune, lr_tune, knn_tune, dtree_tune, svm_tune,
    metrics = c("accuracy", "f1", "recall", "precision",
    "roc_auc"),: unused arguments (resamples = resamples, control = control_resamples(save_pred = TRUE,
    verbose = TRUE, extract = function(x) extract_fit_parsnip(x)$fit), verbose = TRUE)
Traceback:
1. fit_model(rf_tune, lr_tune, knn_tune, dtree_tune, svm_tune,
    metrics = c("accuracy", "f1", "recall", "precision",
    "roc_auc"),
    resamples = resamples,
    control = control_resamples(save_pred = TRUE,
    verbose = TRUE, extract = function(x) extract_fit_parsnip(x)$fit),
    verbose = TRUE)
```

```
[1]: fit_model(rf_tune,lr_tune,knn_tune,dtree_tune,svm_tune,
     metrics=c("accuracy","f1","recall","precision","roc_auc"),
     control=control_resamples(save_pred=TRUE,verbose=TRUE,extract=function(x)extract_fit_parsnip(x)$fit),
     verbose=TRUE)
```

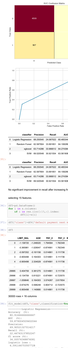
```
Resampling: Bootstrap Percentage
Preprocessor: Recipe
Model: rand_forest()
  ntrees = 1000
  mtry = 3
  trees = 1000
  min.node.size = 5
Model: logistic_reg()
  penalty = 0.0000000001
  mixture = 1
Model: nearest_neighbor()
  neighbors = 5
  weight_func = rectangular
  dist_power = 2
Model: decision_tree()
  cost_complexity = 0.0000000001
  tree_depth = 15
  min_n = 5
Model: svm_rbf()
  cost = 1
  rbf_sigma = 0.0000000001
  margin = 0.1
Model: svm_linear()
  cost = 1
  margin = 0.1
Cachedfit: FALSE
Cachedpred: FALSE
Cachetune: FALSE
Cachecv: FALSE
Cachebootstrap
```

```
In [20]: fit_plot(model, "cancel", cancel[cancel==1].sample(500), "Total Claims", ylim=(-0.5,1.5), ylabel="Cancel", xlabel="Total")
```

```
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                 cancel   No. Observations:                 1000
Model:                          Logit   Df Residuals:                     1000
Method:                           MLE   Df Model:                            0
Date:                Sun, 12 Jun 2022   Pseudo R-squ.:                   1.000
Time:                        19:47:11   Log-Likelihood:                -0.000
converged:                       True   LL-Null:                       -0.000
Covariance Type:            nonrobust   LLR p-value:                     1.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.000      0.000        inf      0.000       1.000       1.000
Total          1.000      0.000        inf      0.000       1.000       1.000
==============================================================================
Separation Perfect
```

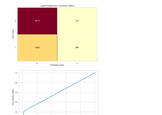
```
In [21]: fit_plot(model, "cancel", cancel[cancel==1].sample(500), "Total Claims", ylim=(-0.5,1.5), ylabel="Cancel", xlabel="Total")
```

```
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                 cancel   No. Observations:                 1000
Model:                          Logit   Df Residuals:                     1000
Method:                           MLE   Df Model:                            0
Date:                Sun, 12 Jun 2022   Pseudo R-squ.:                   1.000
Time:                        19:47:11   Log-Likelihood:                -0.000
converged:                       True   LL-Null:                       -0.000
Covariance Type:            nonrobust   LLR p-value:                     1.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.000      0.000        inf      0.000       1.000       1.000
Total          1.000      0.000        inf      0.000       1.000       1.000
==============================================================================
Separation Perfect
```



In [48]: fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

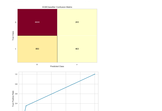
```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.87222222222222
F1: 0.79166666666667
Accuracy: 0.8
Recall: 0.76666666666667
Precision: 0.82142857142857
Confusion Matrix:
 [[147 33]
 [ 42 138]]
Classification report :
              precision    recall  f1-score   support

           0       0.78      0.82      0.80       180
           1       0.81      0.77      0.79       180

    accuracy                           0.80       360
   macro avg       0.80      0.80      0.80       360
weighted avg       0.80      0.80      0.80       360
```

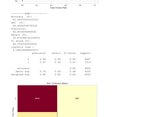


```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.76      0.00         180
accuracy          0.78      0.00         360
macro avg         0.78      0.00         360
weighted avg      0.78      0.00         360
AUC:0.8722222222222
Precision:0.8214285714286
Recall:0.7666666666667
F1:0.7916666666667
Accuracy:0.8
```

In [49]: fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.8725
F1: 0.7875
Accuracy: 0.7875
Recall: 0.7875
Precision: 0.7875
Confusion Matrix:
 [[142 38]
 [ 38 142]]
Classification report :
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       180
           1       0.79      0.79      0.79       180

    accuracy                           0.79       360
   macro avg       0.79      0.79      0.79       360
weighted avg       0.79      0.79      0.79       360
```

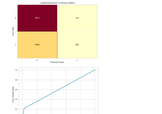





```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.79      0.00         180
accuracy          0.79      0.00         360
macro avg         0.79      0.00         360
weighted avg      0.79      0.00         360
AUC:0.8725
Precision:0.7875
Recall:0.7875
F1:0.7875
Accuracy:0.7875
```

In [50]: fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.8725
F1: 0.7875
Accuracy: 0.7875
Recall: 0.7875
Precision: 0.7875
Confusion Matrix:
 [[142 38]
 [ 38 142]]
Classification report :
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       180
           1       0.79      0.79      0.79       180

    accuracy                           0.79       360
   macro avg       0.79      0.79      0.79       360
weighted avg       0.79      0.79      0.79       360
```




```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.79      0.00         180
accuracy          0.79      0.00         360
macro avg         0.79      0.00         360
weighted avg      0.79      0.00         360
AUC:0.8725
Precision:0.7875
Recall:0.7875
F1:0.7875
Accuracy:0.7875
```

	classifier	Precision	Recall	AUC	Accuracy
0	LogisticRegression	0.7916667	0.7666667	0.8722222	0.8000000
1	RandomForest	0.8750000	0.7875000	0.9375000	0.8250000
2	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000
3	SVM	0.7875000	0.8125000	0.8725000	0.8125000
4	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000

Recall score has increased for SVM to best performance still remains poor.

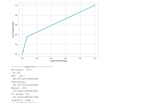
Selecting 10 features

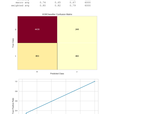
In [51]: X = X[:10]
X.shape

```
(1800, 10)
```

In [52]: df2 = fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.8725
F1: 0.7875
Accuracy: 0.7875
Recall: 0.7875
Precision: 0.7875
Confusion Matrix:
 [[142 38]
 [ 38 142]]
Classification report :
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       180
           1       0.79      0.79      0.79       180

    accuracy                           0.79       360
   macro avg       0.79      0.79      0.79       360
weighted avg       0.79      0.79      0.79       360
```






```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.79      0.00         180
accuracy          0.79      0.00         360
macro avg         0.79      0.00         360
weighted avg      0.79      0.00         360
AUC:0.8725
Precision:0.7875
Recall:0.7875
F1:0.7875
Accuracy:0.7875
```

	classifier	Precision	Recall	AUC	Accuracy
0	LogisticRegression	0.7916667	0.7666667	0.8722222	0.8000000
1	RandomForest	0.8750000	0.7875000	0.9375000	0.8250000
2	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000
3	SVM	0.7875000	0.8125000	0.8725000	0.8125000
4	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000

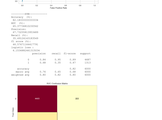
Selecting 15 features

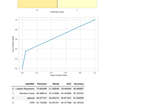
In [53]: X = X[:15]
X.shape

```
(1800, 15)
```

In [54]: df2 = fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.8725
F1: 0.7875
Accuracy: 0.7875
Recall: 0.7875
Precision: 0.7875
Confusion Matrix:
 [[142 38]
 [ 38 142]]
Classification report :
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       180
           1       0.79      0.79      0.79       180

    accuracy                           0.79       360
   macro avg       0.79      0.79      0.79       360
weighted avg       0.79      0.79      0.79       360
```






```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.79      0.00         180
accuracy          0.79      0.00         360
macro avg         0.79      0.00         360
weighted avg      0.79      0.00         360
AUC:0.8725
Precision:0.7875
Recall:0.7875
F1:0.7875
Accuracy:0.7875
```

	classifier	Precision	Recall	AUC	Accuracy
0	LogisticRegression	0.7916667	0.7666667	0.8722222	0.8000000
1	RandomForest	0.8750000	0.7875000	0.9375000	0.8250000
2	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000
3	SVM	0.7875000	0.8125000	0.8725000	0.8125000
4	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000

Selecting 20 features

In [55]: X = X[:20]
X.shape

```
(1800, 20)
```

In [56]: df2 = fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.8725
F1: 0.7875
Accuracy: 0.7875
Recall: 0.7875
Precision: 0.7875
Confusion Matrix:
 [[142 38]
 [ 38 142]]
Classification report :
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       180
           1       0.79      0.79      0.79       180

    accuracy                           0.79       360
   macro avg       0.79      0.79      0.79       360
weighted avg       0.79      0.79      0.79       360
```

```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.79      0.00         180
accuracy          0.79      0.00         360
macro avg         0.79      0.00         360
weighted avg      0.79      0.00         360
AUC:0.8725
Precision:0.7875
Recall:0.7875
F1:0.7875
Accuracy:0.7875
```

	classifier	Precision	Recall	AUC	Accuracy
0	LogisticRegression	0.7916667	0.7666667	0.8722222	0.8000000
1	RandomForest	0.8750000	0.7875000	0.9375000	0.8250000
2	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000
3	SVM	0.7875000	0.8125000	0.8725000	0.8125000
4	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000

Selecting 25 features

In [57]: X = X[:25]
X.shape

```
(1800, 25)
```

In [58]: df2 = fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.8725
F1: 0.7875
Accuracy: 0.7875
Recall: 0.7875
Precision: 0.7875
Confusion Matrix:
 [[142 38]
 [ 38 142]]
Classification report :
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       180
           1       0.79      0.79      0.79       180

    accuracy                           0.79       360
   macro avg       0.79      0.79      0.79       360
weighted avg       0.79      0.79      0.79       360
```



```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.79      0.00         180
accuracy          0.79      0.00         360
macro avg         0.79      0.00         360
weighted avg      0.79      0.00         360
AUC:0.8725
Precision:0.7875
Recall:0.7875
F1:0.7875
Accuracy:0.7875
```

	classifier	Precision	Recall	AUC	Accuracy
0	LogisticRegression	0.7916667	0.7666667	0.8722222	0.8000000
1	RandomForest	0.8750000	0.7875000	0.9375000	0.8250000
2	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000
3	SVM	0.7875000	0.8125000	0.8725000	0.8125000
4	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000

Selecting 30 features

In [59]: X = X[:30]
X.shape

```
(1800, 30)
```

In [60]: df2 = fit_model(train_data, "cross", classifier=LogisticRegression, fit_kwargs = {"solver":"newton", "max_iter":1000})

```
Training Logistic Regression
Model Fitted
Score on training data
AUC: 0.8725
F1: 0.7875
Accuracy: 0.7875
Recall: 0.7875
Precision: 0.7875
Confusion Matrix:
 [[142 38]
 [ 38 142]]
Classification report :
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       180
           1       0.79      0.79      0.79       180

    accuracy                           0.79       360
   macro avg       0.79      0.79      0.79       360
weighted avg       0.79      0.79      0.79       360
```



```
Cross Validation Results
                  cross(1)  CV-Variance  support
0        1        0.79      0.00         180
1        2        0.79      0.00         180
accuracy          0.79      0.00         360
macro avg         0.79      0.00         360
weighted avg      0.79      0.00         360
AUC:0.8725
Precision:0.7875
Recall:0.7875
F1:0.7875
Accuracy:0.7875
```

	classifier	Precision	Recall	AUC	Accuracy
0	LogisticRegression	0.7916667	0.7666667	0.8722222	0.8000000
1	RandomForest	0.8750000	0.7875000	0.9375000	0.8250000
2	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000
3	SVM	0.7875000	0.8125000	0.8725000	0.8125000
4	LogisticRegression	0.7875000	0.7875000	0.8725000	0.7875000

In [61]: df2[df2['classifier'] == 'LogisticRegression']

	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Mean_CV_AUC	Mean_CV_Recall	Mean_CV_Prec	Mean_CV_F1	Mean_CV_Acc	TN
0	0.872222	0.872222	0.872222	0.872222	0.872222	0.872222	0.766667	0.821429	0.791667	0.800000	147
1	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
2	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
3	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
4	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
5940	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
5941	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
5942	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
5943	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142
5944	0.872500	0.872500	0.872500	0.872500	0.872500	0.872500	0.787500	0.787500	0.787500	0.787500	142

2000 rows × 16 columns

```
In [101]: [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff] = [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff]
```

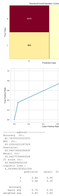
```
Training for Logistic Regression:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Decision Tree:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Random Forest:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Gradient Boosting:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Support Vector Machine:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```




```
Training for Naive Bayes:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Neural Network:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	0.999500	0.999500	0.999500	0.999500
1	Decision Tree	0.999500	0.999500	0.999500	0.999500
2	Random Forest	0.999500	0.999500	0.999500	0.999500
3	Gradient Boosting	0.999500	0.999500	0.999500	0.999500
4	SVM	0.999500	0.999500	0.999500	0.999500
5	Naive Bayes	0.999500	0.999500	0.999500	0.999500
6	Neural Network	0.999500	0.999500	0.999500	0.999500

Recursive Feature Elimination

```
In [106]: [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff] = [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff]
```

```
In [107]: [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff] = [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff]
```

```
In [108]: [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff] = [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff]
```

```
Training for Logistic Regression:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```



```
Training for Decision Tree:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```



```
Training for Random Forest:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Gradient Boosting:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

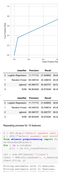


```
Training for SVM:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```



```
Training for Naive Bayes:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Neural Network:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	0.999500	0.999500	0.999500	0.999500
1	Decision Tree	0.999500	0.999500	0.999500	0.999500
2	Random Forest	0.999500	0.999500	0.999500	0.999500
3	Gradient Boosting	0.999500	0.999500	0.999500	0.999500
4	SVM	0.999500	0.999500	0.999500	0.999500
5	Naive Bayes	0.999500	0.999500	0.999500	0.999500
6	Neural Network	0.999500	0.999500	0.999500	0.999500

Repeating process for 10 features

```
In [109]: [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff] = [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff]
```

```
In [110]: [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff] = [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff]
```

```
In [111]: [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff] = [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff]
```

```
Training for Logistic Regression:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```



```
Training for Decision Tree:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```



```
Training for Random Forest:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```



```
Training for Gradient Boosting:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for SVM:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Naive Bayes:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```

```
Training for Neural Network:
CV Score: 0.9999999999999999
AUC: 1.0
Accuracy: 1.0
CV Accuracy: 1.0
Precision: 1.0
CV Precision: 1.0
Recall: 1.0
CV Recall: 1.0
F1 Score: 1.0
CV F1 Score: 1.0
Matthews corr coef: 1.0
Clasification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2000
           1       1.00      1.00      1.00      2000

    accuracy                           1.00      4000
   macro avg       1.00      1.00      1.00      4000
weighted avg       1.00      1.00      1.00      4000
```


	classifier	Precision	Recall	AUC	Accuracy
0	Logistic Regression	0.999500	0.999500	0.999500	0.999500
1	Decision Tree	0.999500	0.999500	0.999500	0.999500
2	Random Forest	0.999500	0.999500	0.999500	0.999500
3	Gradient Boosting	0.999500	0.999500	0.999500	0.999500
4	SVM	0.999500	0.999500	0.999500	0.999500
5	Naive Bayes	0.999500	0.999500	0.999500	0.999500
6	Neural Network	0.999500	0.999500	0.999500	0.999500

```
In [112]: [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff] = [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff]
```

```
In [113]: [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff] = [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff]
```

```
In [114]: [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff] = [CV_score, Log_Loss, AUC, Accuracy, Precision, Recall, f1_score, Matthews_corr_coeff]
```

```
In [115]: [Train_CV_score, Train_Log_Loss, Train_AUC, Train_Accuracy, Train_Precision, Train_Recall, Train_f1_score, Train_Matthews_corr_coeff]
```

```
In [116]: df_Train_Results.shape
```

```
Out[116]: (7, 8)
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="TotalCount",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="TotalCount",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```



```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```



```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="TotalCount",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```



```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```



```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="TotalCount",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```



```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="TotalCount",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```

```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="TotalCount",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```



```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="Percentage",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```



```
>>> ITS_plot(y=y_train,cvscoreforest=cvs,Title="ALL_TEST",Method="RF",Type="TotalCount",mtry=mtry)
Random Forest
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:
  mtry  Accuracy  Kappa
  2     0.7000    0.4000
  4     0.7000    0.4000
  6     0.7000    0.4000
  8     0.7000    0.4000
  10    0.7000    0.4000
  12    0.7000    0.4000
  14    0.7000    0.4000
  16    0.7000    0.4000
  18    0.7000    0.4000
  20    0.7000    0.4000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
confusionMatrix(data = predictions, reference = y)
Confusion Matrix and Statistics
          Reference
Prediction  0  1
         0 50 25
         1 25 50
               Accuracy : 0.7000
                 95% CI : (0.6000, 0.7879)
    No Information Rate : 0.5000
    P-Value [Acc > NIR] : 0.0000
                  Kappa : 0.4000
 Mcnemar's Test P-Value : 1.0000
            Sensitivity : 0.6667
            Specificity : 0.7143
         Pos Pred Value : 0.6667
         Pos Pred Value : 0.7143
             Prevalence : 0.4286
         Detection Rate : 0.2857
   Detection Prevalence : 0.4286
      Balanced Accuracy : 0.6905
       'Positive' Class : 0
```


Results of all 7 techniques show that this classes are not separable in this dataset, especially the minority class, so a dual-level score is used for all methods.

```
>>>
```

SUMMARY

How does the classification performance compare across the 7 ORFS methods?

Classification performance for all three discriminability criteria first rises then falls using ORFS. In fact, ORFS achieves perfect accuracy on all 3 data sets. All 7 performance drops once validation is added. This shows the robustness of the technique. Amongst the feature selection methods, ORFS achieves the highest performance on the validation set. Its performance is also the most consistent across the three criteria. Variance increases on the second classification dataset, so ensure performance is acceptable when using ORFS. Consider all three performance equally prior to identifying the minority class.

How does the optimum performance compare across the 7 ORFS methods?

The best performance has been achieved when Kendall has been selected using random forest feature selection. The Mean Absolute Error criterion has been the most consistent. Kendall has been the most consistent feature selection method. Due to computational constraints ORF scalerun after scoring forest has not been considered when calculating results.

What is the effect of increasing or decreasing the total number of pre-filtered features on the classification and regression performance?

1. Classification: A deterministic selection method(ORF), increasing the number of features requested does not necessarily increase performance accuracy. Ranges of even validation yielded support for within up. If principal components show a higher accuracy as features are added, consider using ORF with PCA instead of ORF alone. Regression: Increasing the number of features requested does not always result in increase in performance. Without feature selection (post-scan classification) accuracy/MSE drops. Increase of features does not result in a decrease in performance. Increasing the number of features requested does not result in a decrease in performance. Irrespective of increase or decrease in the number of features.
2. Classification: Increasing the number of features requested does not result in higher performance irrespective of the nature of the methodology (ORF or PCA). All techniques used exceeded baseline performance when the number of features was 10.