```
# ALL REQUIRED IMPORTS
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from google.colab import drive
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, r2_score, f1_score, accuracy_score, classification_report, roc_au
from sklearn.metrics import *
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import StratifiedKFold, GridSearchCV, KFold, cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.linear_model import *
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from sklearn.ensemble import *
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.decomposition import PCA
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

PCA with Bankruptcy

```
bank_path = '/content/gdrive/MyDrive/Datasets/Bankruptcy/Bankruptcy.csv'
Bankruptcy = pd.read_csv(bank_path)
X = Bankruptcy.drop(['NO','YR','D'], axis=1)
y = Bankruptcy['D']
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y,
                                                    random_state=2022, train_size=0.7)

X_train.shape
```

```
(92, 24)
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_scaled = scaler.transform(X_train)
pca = PCA()
prin_comp = pca.fit_transform(X_scaled)
print(prin_comp.shape)
cols1 = ['PC'+str(i) for i in np.arange(1,len(X.columns)+1)]
pd_PC = pd.DataFrame(prin_comp, columns=cols1)
print(pd_PC)
```

```
(92, 24)
          PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0    2.150639 -1.014643  0.029409 -0.349125  0.613004  0.652186  0.192489
1    4.621861  0.426786 -0.866550 -2.393261  1.391303 -1.425339 -0.365626
2    1.327954 -0.953014 -0.684951  0.038087  0.489552  0.260649 -0.140764
3   -0.862432 -0.235262  0.192424  0.412717  1.409255  0.694081 -0.061490
4    0.090333 -0.960030 -0.246057 -0.765027  1.623973  0.756848 -0.049529
..        ...       ...       ...       ...       ...       ...       ...
87   2.279391 -1.166460 -0.086565  1.322547 -0.639232  0.119697  0.146573
88  -0.071787 -2.251475 -1.758552  1.477135 -0.679139 -0.963157  0.271911
89  -2.411495 -1.049487  0.235811  0.051919  1.466239 -0.614817 -0.122509
90  14.748280  4.624484  4.611596  7.345270  2.289206 -2.883841  0.871485
91  -9.481233  3.943252  2.841515  1.875965  4.458539 -0.943236  0.468542
```

```
         PC8       PC9      PC10  ...      PC15      PC16      PC17      PC18  \
0  -0.157058 -0.990186 -0.178073  ... -0.067100 -0.036103  0.346190  0.121136
1   1.220213  0.217347  0.296993  ... -0.262873  0.283684  0.049705 -0.023219
2  -0.064368 -0.072776 -0.114003  ... -0.013512 -0.065384  0.040599  0.066653
3   0.625637 -0.275590  0.220913  ... -0.114600 -0.128420  0.063672  0.220338
4  -0.842998 -0.499388 -0.022267  ...  0.074763  0.068347  0.290435  0.129448
..       ...       ...       ...  ...       ...       ...       ...       ...
87  0.884234 -0.764814 -0.008651  ... -0.374275 -0.277088  0.084654  0.000212
88  1.044853 -0.899167  0.175668  ... -0.167520  0.197277 -0.256296  0.034021
89 -0.010098  0.225230 -0.117360  ...  0.131543 -0.097198  0.139025  0.069170
90 -2.633785  0.639233 -0.185991  ...  0.071119  0.114973 -0.105719  0.194868
91  2.369074 -0.543670  2.065005  ...  0.643877  0.110610  0.085729  0.038290

        PC19      PC20      PC21      PC22      PC23      PC24
0  -0.058527 -0.105512  0.152431  0.063043 -0.029436  0.001710
1   0.101787 -0.005825  0.004324 -0.001222  0.016491 -0.020664
2  -0.194773  0.006982 -0.015264 -0.002803  0.027096 -0.025739
3   0.046545  0.167568 -0.281270  0.028358 -0.018219 -0.012055
4  -0.116070 -0.004989  0.011963  0.047953 -0.008113 -0.002864
..       ...       ...       ...       ...       ...       ...
87 -0.031834  0.063672 -0.112845  0.024648 -0.003489 -0.012805
88  0.171075  0.149450 -0.013626  0.405562 -0.023550  0.010473
89  0.087966  0.009922 -0.067290  0.047918  0.032776 -0.013467
90 -0.038150  0.006665  0.002766  0.023612  0.005246 -0.001925
91 -0.133500 -0.050573 -0.113700 -0.047739 -0.075851 -0.000947

[92 rows x 24 columns]
```

```python
# With Pipe
pca = PCA()
scaler = StandardScaler()
pipe = Pipeline([('STD', scaler),('PCA',pca)])
prin_comp = pipe.fit_transform(X_train)
print(prin_comp.shape)
cols1 = ['PC'+str(i) for i in np.arange(1,len(X_train.columns)+1)]
pd_PC = pd.DataFrame(prin_comp, columns=cols1)
print(pd_PC)
```

```
(92, 24)
          PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0    2.150639 -1.014643  0.029409 -0.349125  0.613004  0.652186  0.192489
1    4.621861  0.426786 -0.866550 -2.393261  1.391303 -1.425339 -0.365626
2    1.327954 -0.953014 -0.684951  0.038087  0.489552  0.260649 -0.140764
3   -0.862432 -0.235262  0.192424  0.412717  1.409255  0.694081 -0.061490
4    0.090333 -0.960030 -0.246057 -0.765027  1.623973  0.756848 -0.049529
..        ...       ...       ...       ...       ...       ...       ...
87   2.279391 -1.166460 -0.086565  1.322547 -0.639232  0.119697  0.146573
88  -0.071787 -2.251475 -1.758552  1.477135 -0.679139 -0.963157  0.271911
89  -2.411495 -1.049487  0.235811  0.051919  1.466239 -0.614817 -0.122509
90  14.748280  4.624484  4.611596  7.345270  2.289206 -2.883841  0.871485
91  -9.481233  3.943252  2.841515  1.875965  4.458539 -0.943236  0.468542

         PC8       PC9      PC10  ...      PC15      PC16      PC17      PC18  \
0  -0.157058 -0.990186 -0.178073  ... -0.067100 -0.036103  0.346190  0.121136
1   1.220213  0.217347  0.296993  ... -0.262873  0.283684  0.049705 -0.023219
2  -0.064368 -0.072776 -0.114003  ... -0.013512 -0.065384  0.040599  0.066653
3   0.625637 -0.275590  0.220913  ... -0.114600 -0.128420  0.063672  0.220338
4  -0.842998 -0.499388 -0.022267  ...  0.074763  0.068347  0.290435  0.129448
..       ...       ...       ...  ...       ...       ...       ...       ...
87  0.884234 -0.764814 -0.008651  ... -0.374275 -0.277088  0.084654  0.000212
88  1.044853 -0.899167  0.175668  ... -0.167520  0.197277 -0.256296  0.034021
89 -0.010098  0.225230 -0.117360  ...  0.131543 -0.097198  0.139025  0.069170
90 -2.633785  0.639233 -0.185991  ...  0.071119  0.114973 -0.105719  0.194868
91  2.369074 -0.543670  2.065005  ...  0.643877  0.110610  0.085729  0.038290

        PC19      PC20      PC21      PC22      PC23      PC24
0  -0.058527 -0.105512  0.152431  0.063043 -0.029436  0.001710
1   0.101787 -0.005825  0.004324 -0.001222  0.016491 -0.020664
2  -0.194773  0.006982 -0.015264 -0.002803  0.027096 -0.025739
3   0.046545  0.167568 -0.281270  0.028358 -0.018219 -0.012055
4  -0.116070 -0.004989  0.011963  0.047953 -0.008113 -0.002864
```

```
..        ...        ...        ...        ...        ...        ...
87 -0.031834  0.063672 -0.112845  0.024648 -0.003489 -0.012805
88  0.171075  0.149450 -0.013626  0.405562 -0.023550  0.010473
89  0.087966  0.009922 -0.067290  0.047918  0.032776 -0.013467
90 -0.038150  0.006665  0.002766  0.023612  0.005246 -0.001925
91 -0.133500 -0.050573 -0.113700 -0.047739 -0.075851 -0.000947

[92 rows x 24 columns]
```

```
print("The cumulative sum of Ratio of Var of PCs: ",np.cumsum(pca.explained_variance_ratio_*100))
pd_PC = pd.DataFrame(prin_comp, columns=cols1)
```

```
The cumulative sum of Ratio of Var of PCs:  [ 38.84446361  54.47527469  64.6931436   73.4803114   80.84583965
  85.13923795  89.28983752  92.38111189  94.84260262  96.5817658
  97.43720727  98.08885527  98.53871166  98.93680739  99.26314428
  99.49334088  99.65451064  99.78088417  99.8618608   99.91475094
  99.95483964  99.98858825  99.99653303 100.        ]
```

```
svm = SVC(probability = True, random_state = 2022, kernel = 'linear')
pd_PC_trn = pd.DataFrame(prin_comp[:,:8], columns = ['PC'+str(i) for i in np.arange(1,9)] )
svm.fit(pd_PC_trn, y_train)
```

```
SVC(kernel='linear', probability=True, random_state=2022)
```

```
tst_comp = pipe.transform(X_test)
pd_PC_tst = pd.DataFrame(tst_comp[:,:8], columns =['PC'+str(i) for i in np.arange(1,9)])
```

```
y_pred = svm.predict(pd_PC_tst)
print(accuracy_score(y_test, y_pred))
pred_prob = svm.predict_proba(pd_PC_tst)[:,1]
print(roc_auc_score(y_test,pred_prob))
```

```
0.85
0.81
```

Steps:

1. X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, random_state=2022, train_size=0.7)
2. pca = PCA() scaler = StandardScaler() pipe = Pipeline([('STD', scaler),('PCA',pca)])
3. prin_comp = pipe.fit_transform(X_train)
4. pd_PC = pd.DataFrame(prin_comp, columns=cols1)
5. pd_PC_trn = pd.DataFrame(prin_comp[:,:8], columns = ['PC'+str(i) for i in np.arange(1,9)])
6. model.fit(pd_PC_trn)
7. tst_comp = pipe.transform(X_test)
8. pd_PC_tst = pd.DataFrame(tst_comp[:,:8], columns =['PC'+str(i) for i in np.arange(1,9)])
9. y_pred = svm.predict(pd_PC_tst)
10. accuracy_score(y_test, y_pred)

Grid Search with pca in supervised learning

```
scaler = StandardScaler()
prcomp = PCA()
svm = SVC(probability = True, random_state = 2022, kernel ='linear')
pipe_pca_svm = Pipeline([('STD',scaler),('PCA',prcomp),('SVM',svm)])
print(pipe_pca_svm.get_params())
params = { 'PCA__n_components': [0.75, 0.8, 0.85, 0.9, 0.95],
          'SVM__C':[0.4, 1, 2, 2.5]}
kfold = StratifiedKFold(n_splits = 5, shuffle = True, random_state = 2022)
```

```
gcv = GridSearchCV(pipe_pca_svm, param_grid = params,
                   cv = kfold, scoring = 'roc_auc', verbose =3)
gcv.fit(X,y)
print(gcv.best_params_)
print(gcv.best_score_)
```

```
{'memory': None, 'steps': [('STD', StandardScaler()), ('PCA', PCA()), ('SVM', SVC(kernel='linear', probabi
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.863 total time=   0.0s
[CV 2/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.835 total time=   0.0s
[CV 3/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.734 total time=   0.0s
[CV 4/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.970 total time=   0.0s
[CV 5/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.935 total time=   0.0s
[CV 1/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.868 total time=   0.0s
[CV 2/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.846 total time=   0.0s
[CV 3/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.722 total time=   0.0s
[CV 4/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.982 total time=   0.0s
[CV 5/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.929 total time=   0.0s
[CV 1/5] END ..PCA__n_components=0.75, SVM__C=2;, score=0.868 total time=   0.0s
[CV 2/5] END ..PCA__n_components=0.75, SVM__C=2;, score=0.841 total time=   0.0s
[CV 3/5] END ..PCA__n_components=0.75, SVM__C=2;, score=0.704 total time=   0.0s
[CV 4/5] END ..PCA__n_components=0.75, SVM__C=2;, score=0.982 total time=   0.0s
[CV 5/5] END ..PCA__n_components=0.75, SVM__C=2;, score=0.941 total time=   0.0s
[CV 1/5] END PCA__n_components=0.75, SVM__C=2.5;, score=0.868 total time=   0.0s
[CV 2/5] END PCA__n_components=0.75, SVM__C=2.5;, score=0.841 total time=   0.0s
[CV 3/5] END PCA__n_components=0.75, SVM__C=2.5;, score=0.704 total time=   0.0s
[CV 4/5] END PCA__n_components=0.75, SVM__C=2.5;, score=0.982 total time=   0.0s
[CV 5/5] END PCA__n_components=0.75, SVM__C=2.5;, score=0.941 total time=   0.0s
[CV 1/5] END .PCA__n_components=0.8, SVM__C=0.4;, score=0.863 total time=   0.0s
[CV 2/5] END .PCA__n_components=0.8, SVM__C=0.4;, score=0.841 total time=   0.0s
[CV 3/5] END .PCA__n_components=0.8, SVM__C=0.4;, score=0.716 total time=   0.0s
[CV 4/5] END .PCA__n_components=0.8, SVM__C=0.4;, score=0.970 total time=   0.0s
[CV 5/5] END .PCA__n_components=0.8, SVM__C=0.4;, score=0.947 total time=   0.0s
[CV 1/5] END ...PCA__n_components=0.8, SVM__C=1;, score=0.863 total time=   0.0s
[CV 2/5] END ...PCA__n_components=0.8, SVM__C=1;, score=0.835 total time=   0.0s
[CV 3/5] END ...PCA__n_components=0.8, SVM__C=1;, score=0.692 total time=   0.0s
[CV 4/5] END ...PCA__n_components=0.8, SVM__C=1;, score=0.982 total time=   0.0s
[CV 5/5] END ...PCA__n_components=0.8, SVM__C=1;, score=0.941 total time=   0.0s
[CV 1/5] END ...PCA__n_components=0.8, SVM__C=2;, score=0.874 total time=   0.0s
[CV 2/5] END ...PCA__n_components=0.8, SVM__C=2;, score=0.835 total time=   0.0s
[CV 3/5] END ...PCA__n_components=0.8, SVM__C=2;, score=0.692 total time=   0.0s
[CV 4/5] END ...PCA__n_components=0.8, SVM__C=2;, score=0.982 total time=   0.0s
[CV 5/5] END ...PCA__n_components=0.8, SVM__C=2;, score=0.941 total time=   0.0s
[CV 1/5] END .PCA__n_components=0.8, SVM__C=2.5;, score=0.874 total time=   0.0s
[CV 2/5] END .PCA__n_components=0.8, SVM__C=2.5;, score=0.835 total time=   0.0s
[CV 3/5] END .PCA__n_components=0.8, SVM__C=2.5;, score=0.686 total time=   0.0s
[CV 4/5] END .PCA__n_components=0.8, SVM__C=2.5;, score=0.982 total time=   0.0s
[CV 5/5] END .PCA__n_components=0.8, SVM__C=2.5;, score=0.941 total time=   0.0s
[CV 1/5] END PCA__n_components=0.85, SVM__C=0.4;, score=0.874 total time=   0.0s
[CV 2/5] END PCA__n_components=0.85, SVM__C=0.4;, score=0.841 total time=   0.0s
[CV 3/5] END PCA__n_components=0.85, SVM__C=0.4;, score=0.722 total time=   0.0s
[CV 4/5] END PCA__n_components=0.85, SVM__C=0.4;, score=0.953 total time=   0.0s
[CV 5/5] END PCA__n_components=0.85, SVM__C=0.4;, score=0.923 total time=   0.0s
[CV 1/5] END ..PCA__n_components=0.85, SVM__C=1;, score=0.874 total time=   0.0s
[CV 2/5] END ..PCA__n_components=0.85, SVM__C=1;, score=0.835 total time=   0.0s
[CV 3/5] END ..PCA__n_components=0.85, SVM__C=1;, score=0.716 total time=   0.0s
[CV 4/5] END ..PCA__n_components=0.85, SVM__C=1;, score=0.953 total time=   0.0s
[CV 5/5] END ..PCA__n_components=0.85, SVM__C=1;, score=0.923 total time=   0.0s
[CV 1/5] END ..PCA__n_components=0.85, SVM__C=2;, score=0.874 total time=   0.0s
[CV 2/5] END ..PCA__n_components=0.85, SVM__C=2;, score=0.835 total time=   0.0s
[CV 3/5] END ..PCA__n_components=0.85, SVM__C=2;, score=0.704 total time=   0.0s
[CV 4/5] END ..PCA__n_components=0.85, SVM__C=2;, score=0.953 total time=   0.0s
[CV 5/5] END ..PCA__n_components=0.85, SVM__C=2;, score=0.923 total time=   0.0s
```

Results:

1. {'PCA__n_components': 0.75, 'SVM__C': 1}
2. 0.869484361792054

GridSearch CV for PCA on Hr dataset

```
hr_path = '/content/gdrive/MyDrive/Datasets/human-resources-analytics/HR_comma_sep.csv'

hr = pd.read_csv(hr_path)
hr_dum = pd.get_dummies(hr, drop_first=True)
X = hr_dum.drop('left', axis = 1)
y = hr_dum['left']

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, train_size = 0.7, random_state = 2022)

scaler = StandardScaler()
prcomp = PCA()
svm = SVC(probability = True, random_state = 2022, kernel ='linear')
pipe_pca_svm = Pipeline([('STD',scaler),('PCA',prcomp),('SVM',svm)])
print(pipe_pca_svm.get_params())
params = { 'PCA__n_components': [0.75, 0.8, 0.85, 0.9, 0.95],
          'SVM__C':[0.4, 1, 2, 2.5]}
kfold = StratifiedKFold(n_splits = 5, shuffle = True, random_state = 2022)
gcv = GridSearchCV(pipe_pca_svm, param_grid = params,
                   cv = kfold, scoring = 'roc_auc', verbose =3)
gcv.fit(X,y)
print(gcv.best_params_)
print(gcv.best_score_)
```

```
{'memory': None, 'steps': [('STD', StandardScaler()), ('PCA', PCA()), ('SVM',
Fitting 5 folds for each of 20 candidates, totalling 100 fits
[CV 1/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.647 total time=  16
[CV 2/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.725 total time=  17
[CV 3/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.563 total time=  16
[CV 4/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.761 total time=  16
[CV 5/5] END PCA__n_components=0.75, SVM__C=0.4;, score=0.722 total time=  16
[CV 1/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.745 total time=  18
[CV 2/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.680 total time=  19
[CV 3/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.723 total time=  19
[CV 4/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.720 total time=  19
[CV 5/5] END ..PCA__n_components=0.75, SVM__C=1;, score=0.720 total time=  21
[CV 1/5] END ..PCA__n_components=0.75, SVM__C=2;, score=0.737 total time=  22
```

```python

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score


bank_path = '/content/gdrive/MyDrive/Datasets/Bankruptcy/Bankruptcy.csv'
Bankruptcy = pd.read_csv(bank_path)
X = Bankruptcy.drop(['NO','YR','D'], axis=1)
y = Bankruptcy['D']
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y,
                                                    random_state=2022, train_size=0.7)


pca = PCA()
scaler = StandardScaler()

from sklearn.pipeline import Pipeline
pipe = Pipeline([('Scaling',scaler),('PCA', pca)])
prin_comp_trn = pipe.fit_transform(X_train)

cols1 = ['PC'+str(i) for i in np.arange(1,len(X_train.columns)+1)]
print("Variance of the PCA: \n",pca.explained_variance_)
print("The Var Ratio of PCs: \n",pca.explained_variance_ratio_)
print("%age Ratio of Var of PCs: \n",pca.explained_variance_ratio_*100)
print("The cumulative sum of Ratio of Var of PCs: \n",np.cumsum(pca.explained_variance_ratio_*100))

# Silhouette Score
sil = []
for i in np.arange(2,10):
  km = KMeans(n_clusters = i, random_state =2022)
  km.fit(prin_comp_trn)
  labels = km.predict(prin_comp_trn)
  sil.append(silhouette_score(prin_comp_trn, labels))

Ks = np.arange(2,10)
i_max = np.argmax(sil)
best_k = Ks[i_max]

print("Ks: ", Ks,"\nBest K:", best_k)
print(sil)
```

```
Variance of the PCA:
 [9.42511820e+00 3.79261878e+00 2.47923676e+00 2.13209522e+00
 1.78715235e+00 1.04173884e+00 1.00709053e+00 7.50058661e-01
 5.97249618e-01 4.21985967e-01 2.07562063e-01 1.58114151e-01
 1.09151968e-01 9.65928980e-02 7.91815241e-02 5.58542962e-02
 3.91058044e-02 3.06629390e-02 1.96479569e-02 1.28331238e-02
 9.72701738e-03 8.18867385e-03 1.92769979e-03 8.41216436e-04]
The Var Ratio of PCs:
 [3.88444636e-01 1.56308111e-01 1.02178689e-01 8.78716781e-02
 7.36552825e-02 4.29339830e-02 4.15059957e-02 3.09127437e-02
 2.46149072e-02 1.73916318e-02 8.55441475e-03 6.51647994e-03
 4.49856390e-03 3.98095730e-03 3.26336897e-03 2.30196601e-03
 1.61169756e-03 1.26373526e-03 8.09766339e-04 5.28901389e-04
 4.00887039e-04 3.37486105e-04 7.94477721e-05 3.46696991e-05]
%age Ratio of Var of PCs:
```

```
 [3.88444636e+01 1.56308111e+01 1.02178689e+01 8.78716781e+00
 7.36552825e+00 4.29339830e+00 4.15059957e+00 3.09127437e+00
 2.46149072e+00 1.73916318e+00 8.55441475e-01 6.51647994e-01
 4.49856390e-01 3.98095730e-01 3.26336897e-01 2.30196601e-01
 1.61169756e-01 1.26373526e-01 8.09766339e-02 5.28901389e-02
 4.00887039e-02 3.37486105e-02 7.94477721e-03 3.46696991e-03]
The cumulative sum of Ratio of Var of PCs:
 [ 38.84446361  54.47527469  64.6931436   73.4803114   80.84583965
  85.13923795  89.28983752  92.38111189  94.84260262  96.5817658
  97.43720727  98.08885527  98.53871166  98.93680739  99.26314428
  99.49334088  99.65451064  99.78088417  99.8618608   99.91475094
  99.95483964  99.98858825  99.99653303 100.        ]
Ks:  [2 3 4 5 6 7 8 9]
Best K: 5
[0.16937747311428344, 0.20799704734467878, 0.15407019810092057, 0.21541178263367292, 0.17212311794563462, 0.1
```

```
# now we know k = 5 we can use the pipeline for KMeans

scaler = StandardScaler()
km = KMeans(n_clusters = best_k, random_state = 2022)
pipe = Pipeline([('STD',scaler),('KM',km)])
pipe.fit(X_train)
labels = pipe.predict(X_train)
labels
```

```
array([0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       2, 0, 0, 4, 2, 0, 0, 0, 0, 0, 4, 4, 4, 2, 0, 0, 4, 1, 0, 0, 0, 4,
       0, 0, 4, 1, 0, 0, 0, 0, 0, 0, 0, 4, 4, 1, 0, 0, 4, 0, 0, 2, 0, 1,
       0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 0, 0, 4, 1, 4, 2, 0, 0, 0, 4, 4, 0,
       0, 4, 3, 4], dtype=int32)
```

```
X_train['Cluster'] = labels
X_train
```

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	...	R16	R17	R
42	0.03	0.01	0.01	0.02	-0.04	-0.07	-0.10	6.88	1.10	0.36	...	0.00	0.00	0.
32	0.07	0.04	0.05	0.06	0.04	0.05	0.06	2.68	1.19	0.66	...	-0.16	-0.21	-0.
35	0.06	0.02	0.02	0.03	0.01	0.02	0.03	3.45	1.56	0.47	...	0.02	0.02	0.
110	0.15	0.02	0.03	0.10	0.03	0.04	0.11	3.76	2.55	0.42	...	0.06	0.08	0.
122	0.02	0.01	0.01	0.02	0.00	0.00	-0.01	2.92	1.64	0.52	...	0.06	0.09	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
61	0.10	0.02	0.02	0.04	-0.01	-0.01	-0.02	8.91	1.17	0.24	...	-0.01	-0.01	-0.
6	0.08	0.04	0.02	0.03	0.30	0.13	0.21	3.72	0.82	0.42	...	0.15	0.06	0.
98	0.12	0.02	0.04	0.09	0.11	0.17	0.40	2.22	2.11	0.43	...	0.10	0.16	0.
43	0.04	0.02	0.01	0.01	-1.40	-0.42	-0.30	0.35	0.27	0.19	...	-0.59	-0.18	-0.
103	0.70	0.08	0.11	0.70	0.10	0.15	0.96	1.74	4.64	0.50	...	0.14	0.20	1.

92 rows × 25 columns

```
X_train['Cluster'] = X_train['Cluster'].astype('category')
```

```
X_train['Cluster'].value_counts()
```

```
0    61
4    18
1     6
2     6
3     1
Name: Cluster, dtype: int64
```

```
X_trn_ohe = pd.get_dummies(X_train)
labels=pipe.predict(X_test)
print(labels)
print(X_trn_ohe)
```

```
[0 4 0 0 0 0 4 0 1 4 4 0 0 0 4 2 0 0 0 0 0 4 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0
 0 0 0]
       R1    R2    R3    R4    R5    R6    R7    R8    R9   R10  ...   R20  \
42   0.03  0.01  0.01  0.02 -0.04 -0.07 -0.10  6.88  1.10  0.36  ...  1.60
32   0.07  0.04  0.05  0.06  0.04  0.05  0.06  2.68  1.19  0.66  ...  1.28
35   0.06  0.02  0.02  0.03  0.01  0.02  0.03  3.45  1.56  0.47  ...  1.30
110  0.15  0.02  0.03  0.10  0.03  0.04  0.11  3.76  2.55  0.42  ...  1.43
122  0.02  0.01  0.01  0.02  0.00  0.00 -0.01  2.92  1.64  0.52  ...  1.51
..    ...   ...   ...   ...   ...   ...   ...   ...   ...   ...  ...   ...
61   0.10  0.02  0.02  0.04 -0.01 -0.01 -0.02  8.91  1.17  0.24  ...  1.15
6    0.08  0.04  0.02  0.03  0.30  0.13  0.21  3.72  0.82  0.42  ...  0.42
98   0.12  0.02  0.04  0.09  0.11  0.17  0.40  2.22  2.11  0.43  ...  1.57
43   0.04  0.02  0.01  0.01 -1.40 -0.42 -0.30  0.35  0.27  0.19  ...  0.30
103  0.70  0.08  0.11  0.70  0.10  0.15  0.96  1.74  4.64  0.50  ...  1.42

      R21   R22   R23   R24  Cluster_0  Cluster_1  Cluster_2  Cluster_3  \
42   1.46  0.00  0.00  0.00          1          0          0          0
32   1.36 -0.16 -0.21 -0.28          0          0          1          0
35   1.41  0.03  0.03  0.05          1          0          0          0
110  2.94  0.06  0.08  0.23          1          0          0          0
122  1.79  0.06  0.09  0.16          1          0          0          0
..    ...   ...   ...   ...        ...        ...        ...        ...
61   1.74 -0.01 -0.01 -0.02          1          0          0          0
6    1.67  0.12  0.05  0.09          1          0          0          0
98   2.36  0.10  0.16  0.37          0          0          0          0
43   0.71 -1.49 -0.45 -0.32          0          0          0          1
103  6.50  0.14  0.20  1.31          0          0          0          0

     Cluster_4
42           0
32           0
35           0
110          0
122          0
..         ...
61           0
6            0
98           1
43           0
103          1

[92 rows x 29 columns]
```

```
from sklearn.ensemble import RandomForestClassifier
X_test['Cluster'] = labels
X_test['Cluster'] = X_test['Cluster'].astype('category')
X_tst_ohe = pd.get_dummies(X_test)

X_tst_ohe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 40 entries, 115 to 33
Data columns (total 28 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   R1         40 non-null     float64
 1   R2         40 non-null     float64
 2   R3         40 non-null     float64
 3   R4         40 non-null     float64
 4   R5         40 non-null     float64
 5   R6         40 non-null     float64
 6   R7         40 non-null     float64
 7   R8         40 non-null     float64
 8   R9         40 non-null     float64
 9   R10        40 non-null     float64
 10  R11        40 non-null     float64
```

```
 11  R12        40 non-null     float64
 12  R13        40 non-null     float64
 13  R14        40 non-null     float64
 14  R15        40 non-null     float64
 15  R16        40 non-null     float64
 16  R17        40 non-null     float64
 17  R18        40 non-null     float64
 18  R19        40 non-null     float64
 19  R20        40 non-null     float64
 20  R21        40 non-null     float64
 21  R22        40 non-null     float64
 22  R23        40 non-null     float64
 23  R24        40 non-null     float64
 24  Cluster_0  40 non-null     uint8
 25  Cluster_1  40 non-null     uint8
 26  Cluster_2  40 non-null     uint8
 27  Cluster_4  40 non-null     uint8
dtypes: float64(24), uint8(4)
memory usage: 8.0 KB
```

```
X_trn_ohe.drop('Cluster_3', axis =1, inplace = True)
rf = RandomForestClassifier(random_state = 2022)
rf.fit(X_trn_ohe, y_train)

y_pred = rf.predict(X_tst_ohe)
print(accuracy_score(y_test, y_pred))

y_pred_prob = rf.predict_proba(X_tst_ohe)[:,1]
print(roc_auc_score(y_test, y_pred_prob))
```

```
0.875
0.93625
```

Double-click (or enter) to edit