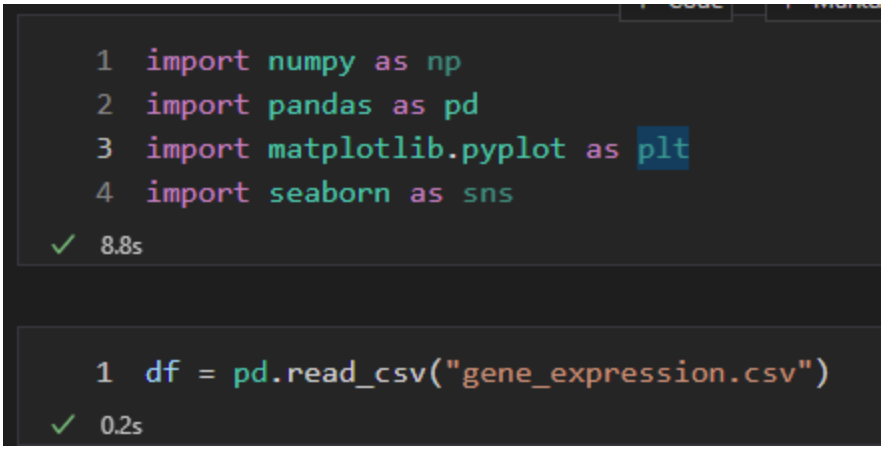
K Nearest Neigbors



▼ Data Structure

- Data & Scatter

```
1 df.head()
```

✓ 0.4s Pyth

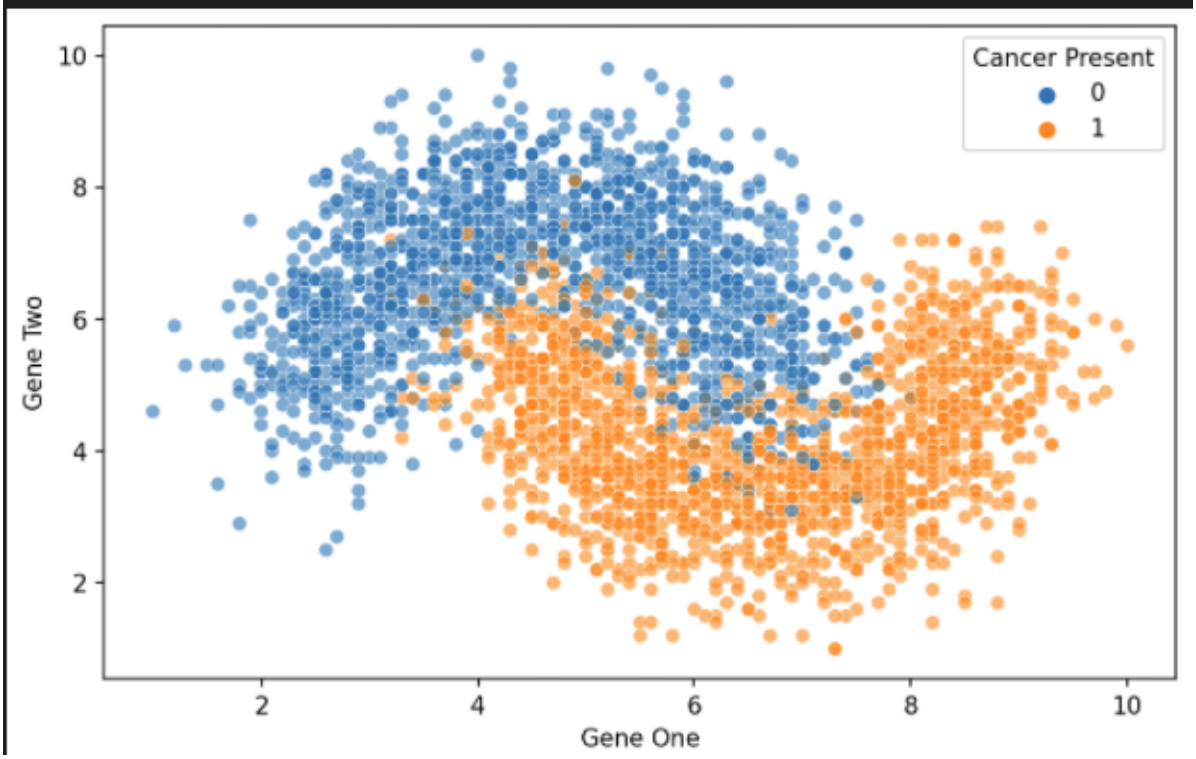
	Gene One	Gene Two	Cancer Present
0	4.3	3.9	1
1	2.5	6.3	0
2	5.7	3.9	1
3	6.1	6.2	0
4	7.4	3.4	1

```
1 X = df.drop("Cancer Present", axis= 1)
2 y = df["Cancer Present"]
```

✓ 0.1s Pyth

```
1 plt.figure(figsize=(8,5),dpi=150)
2 sns.scatterplot(data=df, x="Gene One", y="Gene Two",
3 hue="Cancer Present", alpha=0.6);
```

✓ 0.5s Pyth

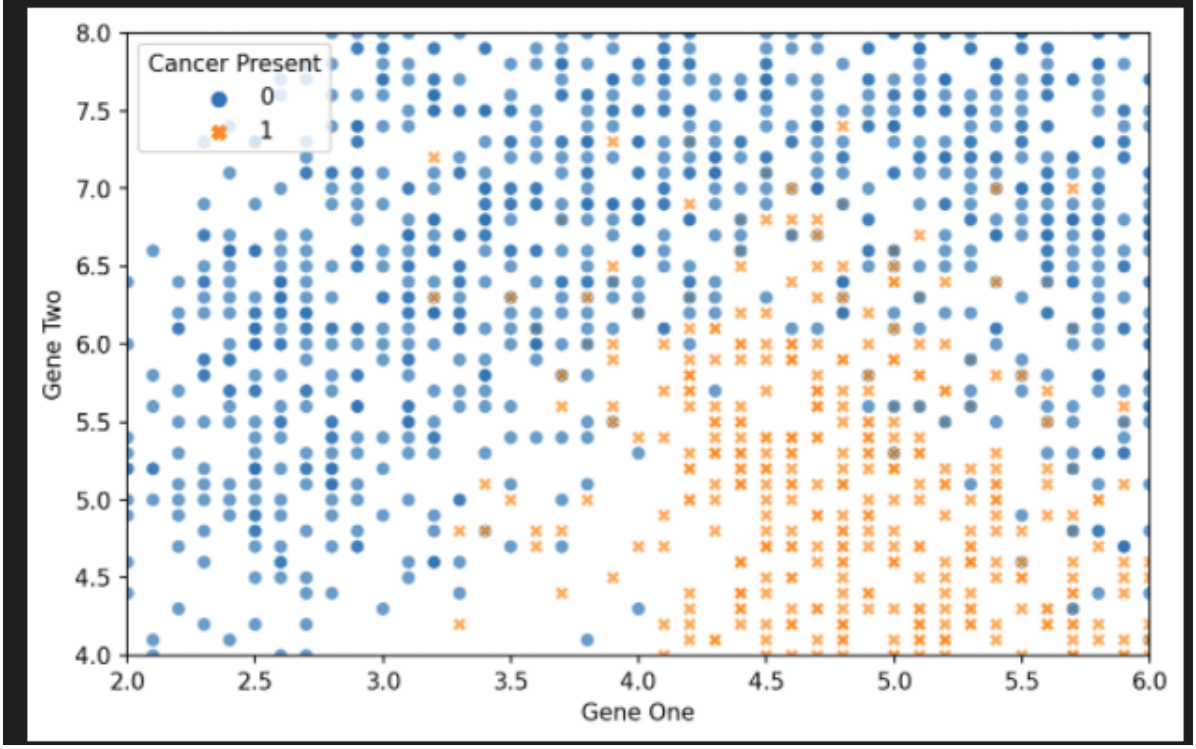


- zoomed Plot
  style="Cancer Present" : cancerr presetnt olanları x ve o olarak ayırır.

```
1  plt.figure(figsize=(8,5),dpi=150)
2  sns.scatterplot(data=df, x="Gene One", y="Gene Two",
3  | hue="Cancer Present", alpha=0.7, style="Cancer Present")
4  plt.xlim(2,6)
5  plt.ylim(4,8)
```

✓ 0.8s Pytho

```
(4.0, 8.0)
```



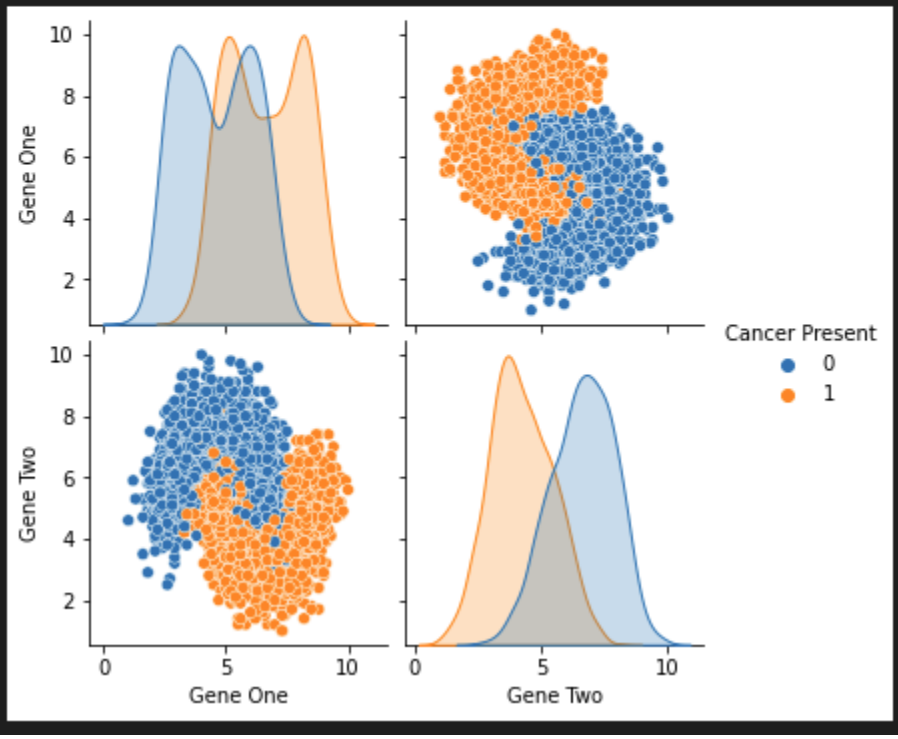
- pair plot

```
1  plt.figure(figsize=(8,5),dpi=150)
2  sns.pairplot(data=df, hue="Cancer Present")
```

✓ 4.3s

```
<seaborn.axisgrid.PairGrid at 0x1f10db485e0>

<Figure size 1200x750 with 0 Axes>
```



▼ Model Preparing

- scalerization

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```
✓ 0.9s

```python
test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```
✓ 0.9s

```python
scaler = StandardScaler()
```
✓ 0.1s

```python
scaled_X_Train = scaler.fit_transform(X_train)
scaled_X_Test = scaler.transform(X_test)
```
✓ 0.5s

- KNN Model

```python
from sklearn.neighbors import KNeighborsClassifier
```
✓ 0.5s

```python
knn_model = KNeighborsClassifier(n_neighbors=1)
```
✓ 0.4s

```python
knn_model.fit(scaled_X_Train,y_train)
```
✓ 0.1s

```
KNeighborsClassifier(n_neighbors=1)
```

```python
y_pred = knn_model.predict(scaled_X_Test)
```
✓ 0.6s

- accuracy and Results

```
1 from sklearn.metrics import classification_report, confusion_matrix
```
✓ 0.1s

```
1 confusion_matrix(y_test,y_pred)
```
✓ 0.8s

```
array([[420,  50],
       [ 47, 383]], dtype=int64)
```

```
1 print(classification_report(y_test, y_pred))
```
✓ 0.2s

```
              precision    recall  f1-score   support

           0       0.90      0.89      0.90       470
           1       0.88      0.89      0.89       430

    accuracy                           0.89       900
   macro avg       0.89      0.89      0.89       900
weighted avg       0.89      0.89      0.89       900
```

▼ Choosing K

- accuracy and import

```
1 from sklearn.metrics import accuracy_score
```
✓ 0.5s

```
1 1 - accuracy_score(y_test,y_pred)
```
✓ 0.1s

```
0.10777777777777775
```

- K Choosing Func

```
test_error_rate =[]

for k in range(1,30):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(scaled_X_Train,y_train)

    y_pred_test = knn_model.predict(scaled_X_Test)

    test_error = 1 - accuracy_score(y_test, y_pred_test)
    test_error_rate.append(test_error)
```

✓ 1.5s

```
test_error_rate
```

✓ 0.5s

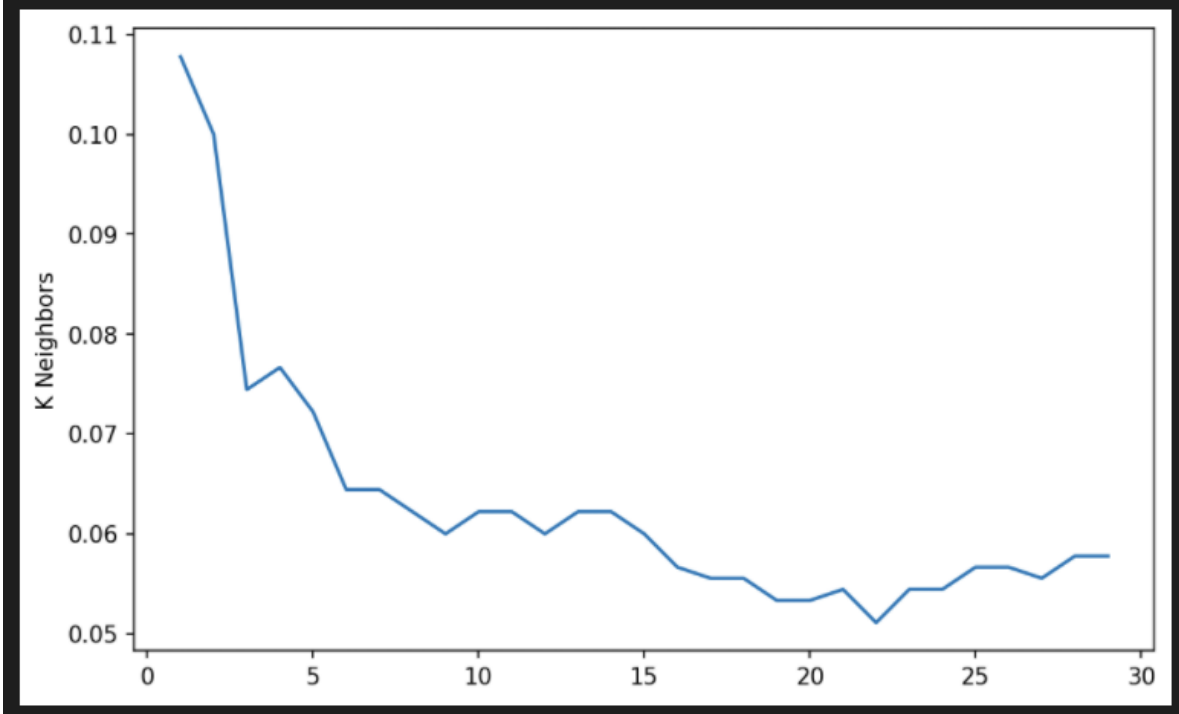
```
Output exceeds the size limit. Open the full output data in a text editor
[0.10777777777777775,
 0.09999999999999998,
 0.07444444444444442,
 0.07666666666666666,
 0.07222222222222219,
 0.06444444444444442,
 0.06444444444444442,
 0.06222222222222218,
 0.06000000000000005,
 0.06222222222222218,
 0.06222222222222218,
 0.06000000000000005,
 0.06222222222222218,
 0.06222222222222218,
 0.06000000000000005,
...
```

- Elbow (Best K neighbor num)

```python
plt.figure(figsize=(8,5),dpi=150)
plt.plot(range(1,30),test_error_rate)
plt.ylabel("ERROR RATE")
plt.ylabel("K Neighbors")
```

✓ 0.1s Python

```
Text(0, 0.5, 'K Neighbors')
```



▼ Pipeline

- scalarization and parameters

```
1  scaler = StandardScaler()
```

```
1  knn = KNeighborsClassifier()
```
✓ 0.3s

```
1  knn.get_params().keys()
```
✓ 0.1s

```
dict_keys(['algorithm', 'leaf_size', 'metric', 'metric_params', 'n_jobs',
'n_neighbors', 'p', 'weights'])
```

```
1  operations = [("scaler",scaler),("knn",knn)]
```
✓ 0.3s

- Grid Search cv

```
1  from sklearn.pipeline import Pipeline
```
✓ 0.2s

```
1  pipe = Pipeline(operations)
```
✓ 0.8s

+ Code + Markdown

```
1  from sklearn.model_selection import GridSearchCV
```
✓ 0.3s

```
1  k_values = list(range(1,20))
2  k_values
```
✓ 0.1s

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
1  param_grid = {'knn__n_neighbors': k_values}
```
✓ 0.6s

```
1  full_cv_classifier = GridSearchCV(pipe, param_grid, cv=5, scoring="accuracy")
```
✓ 0.1s

```
1  full_cv_classifier.fit(X_train,y_train)
```
✓ 2.9s

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('knn', KNeighborsClassifier())]),
             param_grid={'knn__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                              12, 13, 14, 15, 16, 17, 18, 19]},
             scoring='accuracy')
```

- best parameters

```
1 full_cv_classifier.best_estimator_.get_params()
```

✓ 0.9s

```
{'memory': None,
 'steps': [('scaler', StandardScaler()),
  ('knn', KNeighborsClassifier(n_neighbors=14))],
 'verbose': False,
 'scaler': StandardScaler(),
 'knn': KNeighborsClassifier(n_neighbors=14),
 'scaler__copy': True,
 'scaler__with_mean': True,
 'scaler__with_std': True,
 'knn__algorithm': 'auto',
 'knn__leaf_size': 30,
 'knn__metric': 'minkowski',
 'knn__metric_params': None,
 'knn__n_jobs': None,
 'knn__n_neighbors': 14,
 'knn__p': 2,
 'knn__weights': 'uniform'}
```

```
1 full_pred = full_cv_classifier.predict(X_test)
```

✓ 0.1s

```
1 print(classification_report(y_test,full_pred))
```

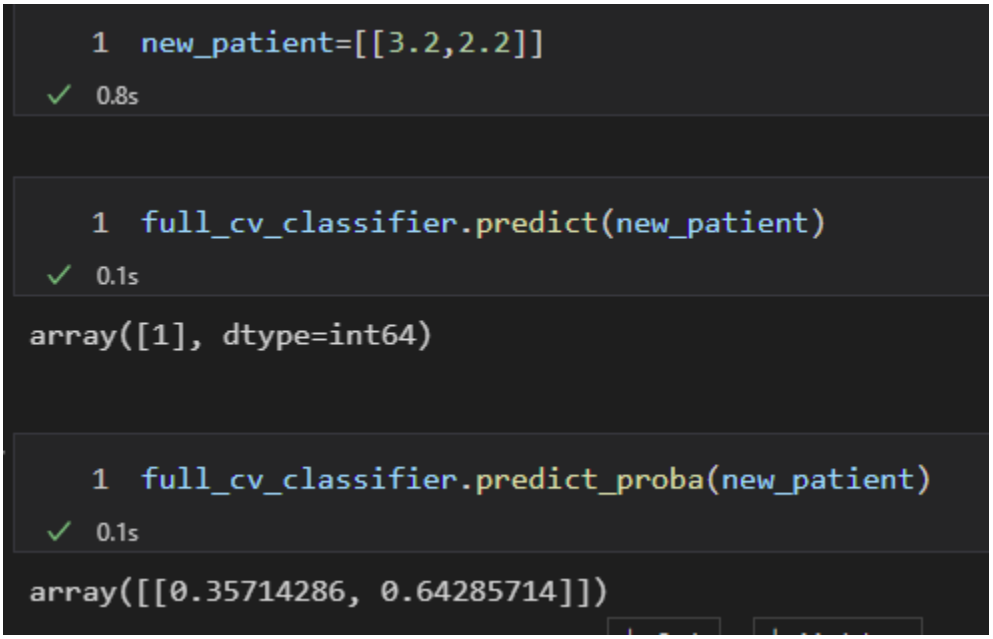
✓ 0.9s

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       470
           1       0.95      0.92      0.93       430

    accuracy                           0.94       900
   macro avg       0.94      0.94      0.94       900
weighted avg       0.94      0.94      0.94       900
```

- Predicted Example



▼ Project

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

6.2s Python

```python
df = pd.read_csv('sonar.all-data.csv')
```

0.7s Python

Data and Viz

```python
df.head()
```

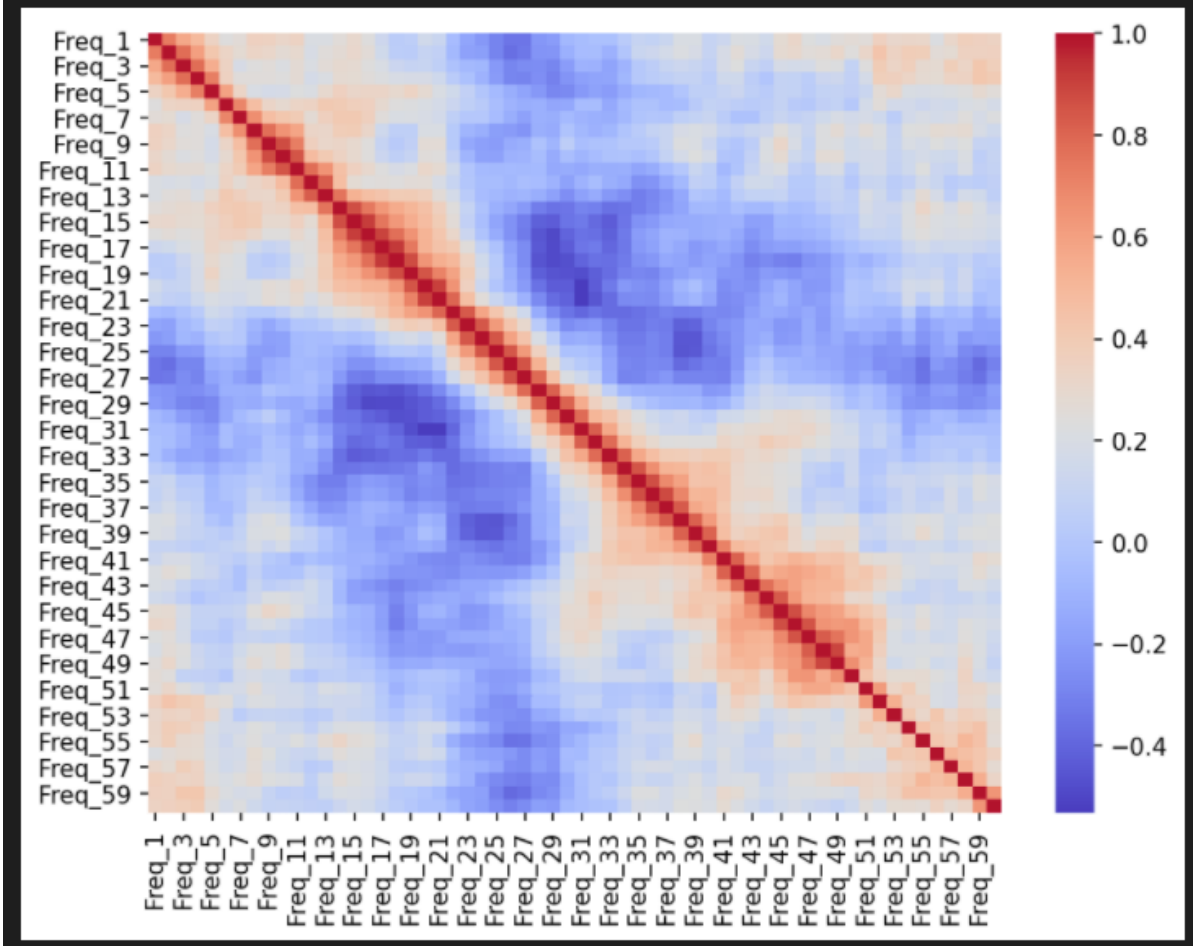
0.8s Python

	Freq_1	Freq_2	Freq_3	Freq_4	Freq_5	Freq_6	Freq_7	Freq_8	Freq_9	Freq_10	...	F
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	

```
1  plt.figure(figsize=(8,6), dpi=150)
2  sns.heatmap(df.corr(), cmap="coolwarm")
```

✓ 1.5s Pytho

```
<AxesSubplot:>
```



```
1  df["Target"] = df["Label"].map({"R":0,"M":1})
```

✓ 0.4s Python

```
1  df.head()
```

✓ 0.6s Python

	Freq_1	Freq_2	Freq_3	Freq_4	Freq_5	Freq_6	Freq_7	Freq_8	Freq_9	Freq_10	...
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...

5 rows × 62 columns

```
1  df.corr()["Target"].sort_values()
```

✓ 0.6s

```
Freq_36   -0.269151
Freq_35   -0.227670
Freq_37   -0.209055
Freq_34   -0.172010
Freq_31   -0.110728
             ...
Freq_10    0.341142
Freq_49    0.351312
Freq_12    0.392245
Freq_11    0.432855
Target     1.000000
Name: Target, Length: 61, dtype: float64
```

```
1  from sklearn.model_selection import train_test_split
```
✓ 0.6s

```
1  X=df.drop(["Target","Label"], axis=1)
2  y=df["Label"]
```
✓ 0.4s

```
1  X_cv, X_test, y_cv, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
```
✓ 0.5s

```
1  from sklearn.neighbors import KNeighborsClassifier
2  from sklearn.preprocessing import StandardScaler
```
✓ 0.4s

```
1  scaler = StandardScaler()
```
✓ 0.5s

```
1  knn = KNeighborsClassifier()
```
✓ 0.5s

```
1  operations = [("scaler",scaler),("knn",knn)]
```
✓ 0.3s

```
1  from sklearn.pipeline import Pipeline
```
✓ 0.4s

```
1  pipe = Pipeline(operations)
```
✓ 0.3s

```
1  from sklearn.model_selection import GridSearchCV
```
✓ 0.4s

```
1  k_values = list(range(1,30))
```
✓ 0.4s

```
1  param_grid = {"knn__n_neighbors" : k_values}
```
✓ 0.6s

```
1  full_cv_classifier = GridSearchCV(pipe,param_grid,cv=5,scoring="accuracy" )
```
✓ 0.1s

```
1  full_cv_classifier.fit(X_cv,y_cv)
```
✓ 1.9s

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('knn', KNeighborsClassifier())]),
             param_grid={'knn__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                              12, 13, 14, 15, 16, 17, 18, 19,
                                              20, 21, 22, 23, 24, 25, 26, 27,
                                              28, 29]},
             scoring='accuracy')
```

- en iyi parametre değerleri

```
1  full_cv_classifier.best_estimator_.get_params()
```

✓ 0.4s

```
{'memory': None,
 'steps': [('scaler', StandardScaler()),
  ('knn', KNeighborsClassifier(n_neighbors=1))],
 'verbose': False,
 'scaler': StandardScaler(),
 'knn': KNeighborsClassifier(n_neighbors=1),
 'scaler__copy': True,
 'scaler__with_mean': True,
 'scaler__with_std': True,
 'knn__algorithm': 'auto',
 'knn__leaf_size': 30,
 'knn__metric': 'minkowski',
 'knn__metric_params': None,
 'knn__n_jobs': None,
 'knn__n_neighbors': 1,
 'knn__p': 2,
 'knn__weights': 'uniform'}
```

- sonuçları DF'ye çevirme

```
1 pd.DataFrame(full_cv_classifier.cv_results_)
```

0.9s

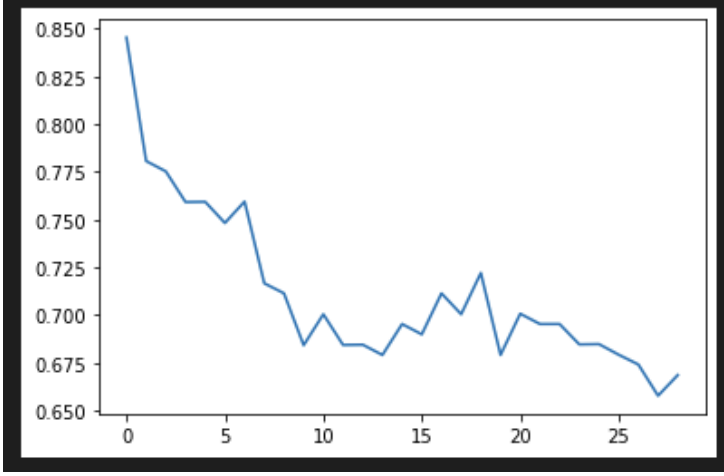
	mean_fit_time	std_fit_time	mean_score_time	std_score_time
0	0.008377	0.000489	0.007380	0.004351
1	0.006981	0.001092	0.005585	0.001018
2	0.005784	0.001466	0.003989	0.000892
3	0.006981	0.001093	0.004788	0.000977
4	0.004987	0.001093	0.004588	0.001353

- ortalama test sonucu plotu

```
1 )ataFrame(full_cv_classifier.cv_results_)["mean_test_score"].plot()
```

0.2s Pytho

```
<AxesSubplot:>
```

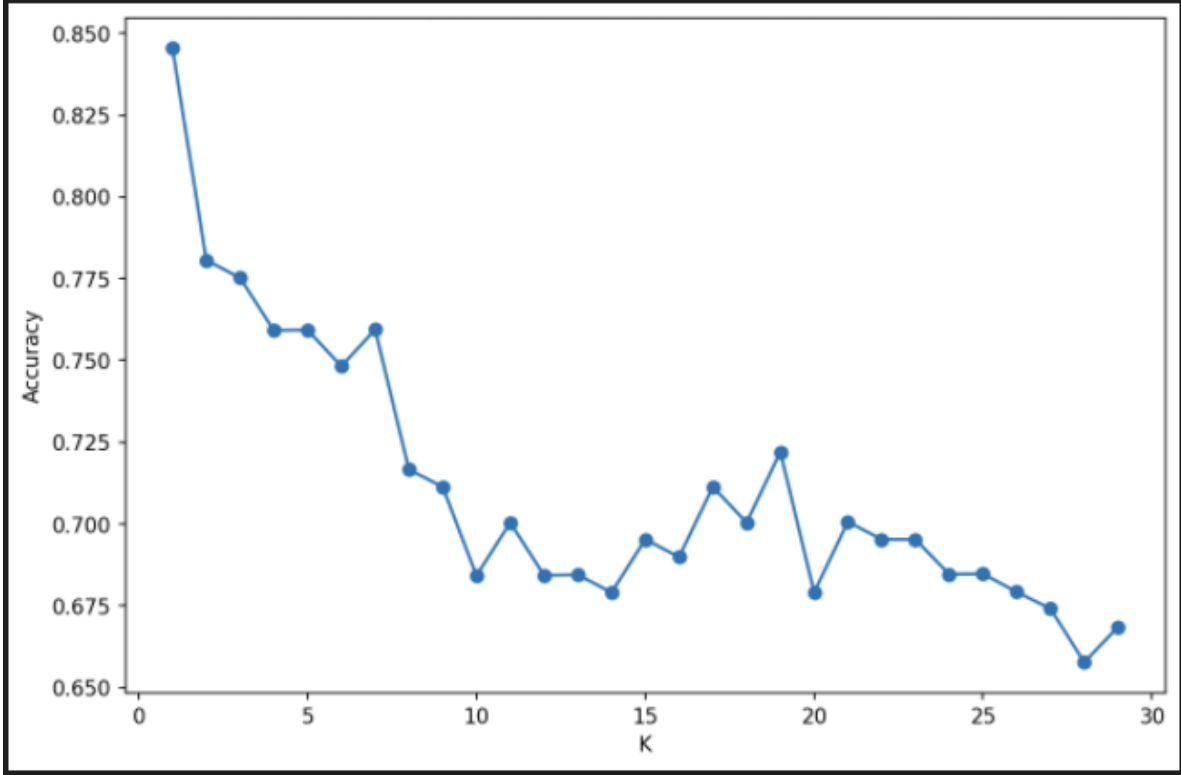


- aynısı ama daha şeqil

```
1  plt.figure(figsize=(9,6), dpi=150)
2  scores = full_cv_classifier.cv_results_['mean_test_score']
3  plt.plot(k_values,scores,'o-')
4  plt.xlabel("K")
5  plt.ylabel("Accuracy")
```

✓ 0.2s Python

```
Text(0, 0.5, 'Accuracy')
```



Evaluation

```
1  y_pred = full_cv_classifier.predict(X_test)
```

✓ 0.3s Pytho

```
1  from sklearn.metrics import classification_report,confusion_matrix
```

✓ 0.3s Pytho

```
1  confusion_matrix(y_test,y_pred)
```

✓ 0.9s Pytho

```
array([[12,  1],
       [ 1,  7]], dtype=int64)
```

```
1  print(classification_report(y_test,y_pred))
```

✓ 0.9s Pytho

```
              precision    recall  f1-score   support

           M       0.92      0.92      0.92        13
           R       0.88      0.88      0.88         8

    accuracy                           0.90        21
   macro avg       0.90      0.90      0.90        21
weighted avg       0.90      0.90      0.90        21
```