```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

K- უახლოესი მეზობლების მეთოდი (KNN - K Nearest Neighbors) K- უახლოესი მეზობლის მეთოდი გამოიყენება მონაცემთა კლასიფიკაციისთვის. ამ ლექციაში ჩვენ ვიმუშავებთ გენების მონაცემთა ბაზასთან. გენის დონეები გამოითვლება როგორც თანაფარდობა სამიზნე გენის დონესა (შესწავლილი გენი) და ერთი ან მეტი საცნობარო გენის დონეს შორის (ჩვეულებრივ, არსებული გენები). ეს მონაცემთა ნაკრები არის სინთეზური და სპეციალურად შეიქმნა კლასიფიკაციის KNN მეთოდის ძლიერი და სუსტი მხარეების საჩვენებლად.

გენების შესახებ მეტი შეგიძლიათ წაიკითხოთ აქ: https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/gene-expression-level

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

მონაცემები

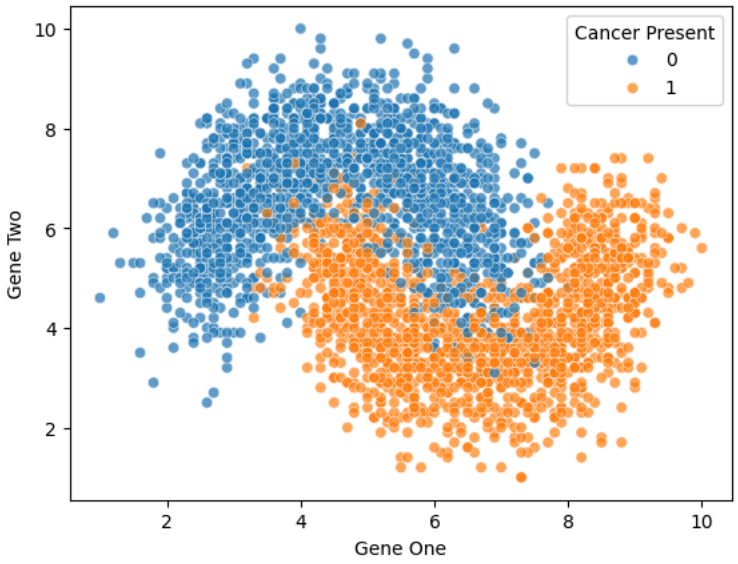
```
1 df = pd.read_csv('/content/drive/MyDrive/Dataset/gene_expression.csv')
```

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

	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

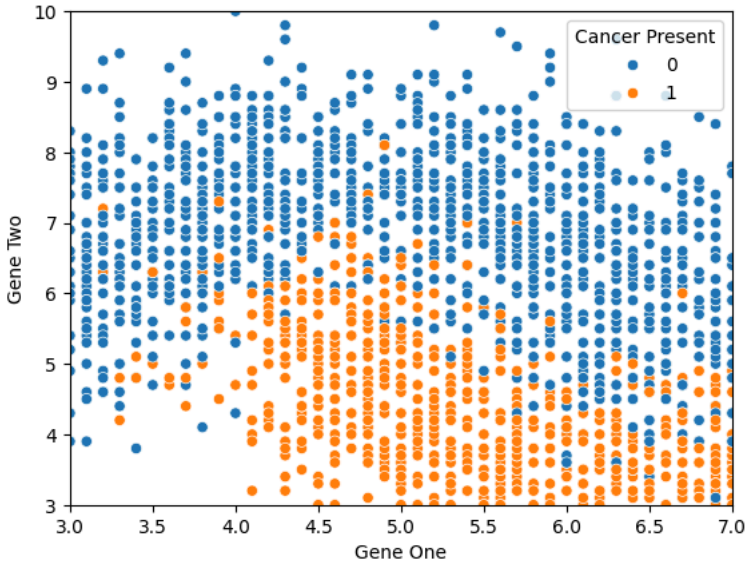
```
1 sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',data=df,alpha=0.7)
```

```
<Axes: xlabel='Gene One', ylabel='Gene Two'>
```




```
1 sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',data=df, alpha=1)
2 plt.xlim(3,7)
3 plt.ylim(3,10)
4 #plt.legend(loc=(1.1,0.5))
```

```
(3.0, 10.0)
```

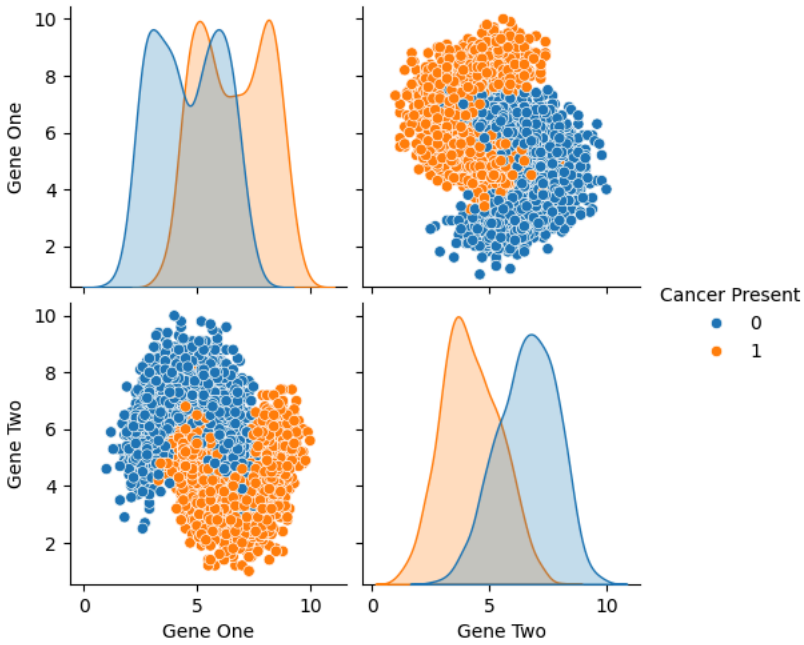



```
1 len(df)
```

```
3000
```

```
1 sns.pairplot(data=df, hue='Cancer Present' )
```

```
<seaborn.axisgrid.PairGrid at 0x7a6f84fdd600>
```



სასწავლო და სატესტო მონაცემთა დაყოფა, ასევე მონაცემთა მასშტაბირება (სკალირება)

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.preprocessing import StandardScaler
```

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

	Gene One	Gene Two
0	4.3	3.9
1	2.5	6.3
2	5.7	3.9
3	6.1	6.2
4	7.4	3.4
...	...	...
2995	5.0	6.5
2996	3.4	6.6
2997	2.7	6.5
2998	3.3	5.6
2999	4.6	8.2

3000 rows × 2 columns

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

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

```
1 scaled_X_train = scaler.fit_transform(X_train)
2 scaled_X_test = scaler.transform(X_test)
```

```
1 from sklearn.neighbors import KNeighborsClassifier
```

```
1 #help(KNeighborsClassifier)
```

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

```
1 knn_model.fit(scaled_X_train,y_train)
```

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

```
1 y_pred = knn_model.predict(scaled_X_test)
```

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

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

```
array([[422,  48],
       [ 50, 380]])
```

```
1 len(y_test)
```

```
900
```

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

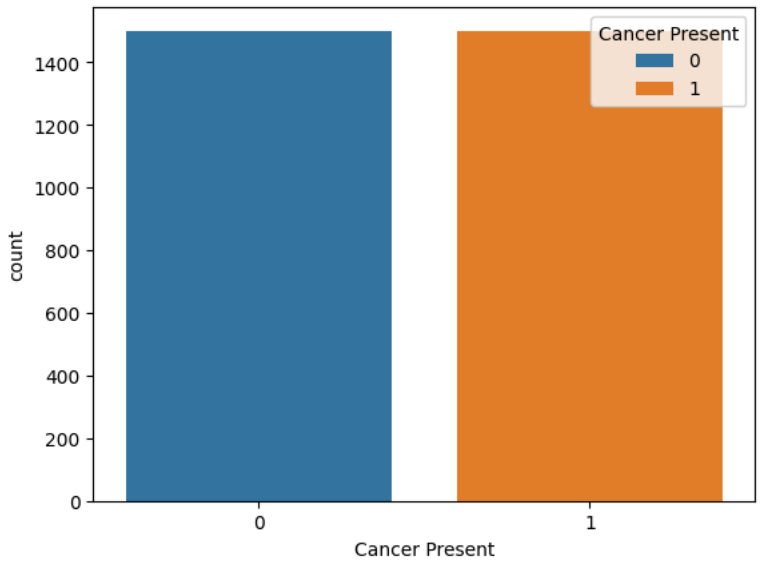
```
                  precision    recall  f1-score   support

               0       0.89      0.90      0.90       470
               1       0.89      0.88      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
```

```
1 sns.countplot(x='Cancer Present', hue="Cancer Present", data=df)
```

```
<Axes: xlabel='Cancer Present', ylabel='count'>
```




```
1 df["Cancer Present"].value_counts()
```

```
1    1500
0    1500
Name: Cancer Present, dtype: int64
```

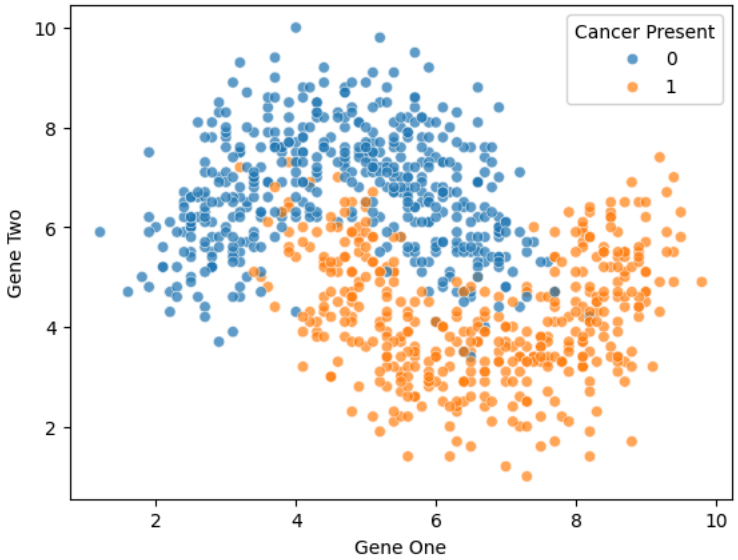
```
1 full_test = pd.concat([X_test,y_test],axis=1)
```

```
1 len(full_test)
```

```
900
```

```
1 sns.scatterplot(x='Gene One',y='Gene Two',hue='Cancer Present',
2                 data=full_test,alpha=0.7)
```

```
<Axes: xlabel='Gene One', ylabel='Gene Two'>
```



მოდელის შეფასება

```
1 y_pred = knn_model.predict(scaled_X_test)
```

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

```
1 accuracy_score(y_test,y_pred)
```

```
0.8911111111111111
```

KNN-ის ინტერპრეტაცია და K-ის მნიშვნელობის არჩევა.

elbow იდაყვის მეთოდი

კარგი K მნიშვნელობების არჩევისთვის შენიშვნა: აქ ჩვენ ვიყენებთ სატესტო მონაცემთა ნაკრების ასარჩევად ჰიპერპარამეტრს K.

```
test_error_rates = []


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_rates.append(test_error)
```

უმეტეს ამოცანების შემთხვევაში (ყველა ამოცანებში არა) შეამჩნევთ, რომ K მნიშვნელობის გაზრდა იწვევს შეცდომების შემცირებას, თუმცა რაღაც მომენტში შეცდომების შემცირება ირღვევა. ზოგჯერ, პირიქითაც ხდება შეცდომები იზრება, როდესაც K-ს მნიშვნელობა ძალიან დიდია.

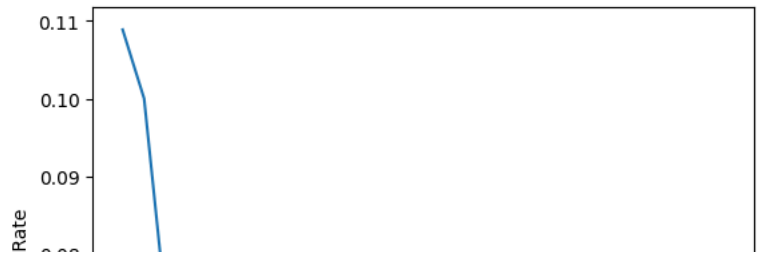
```
test_error_rates
```

```
[0.10888888888888892,
 0.09999999999999998,
 0.07333333333333336,
 0.0755555555555556,
 0.07222222222222219,
 0.06666666666666665,
 0.06444444444444442,
 0.06444444444444442,
 0.05777777777777782,
 0.06333333333333335,
 0.061111111111111116,
 0.06000000000000005,
 0.061111111111111116,
 0.06222222222222218,
 0.05888888888888888,
 0.05777777777777782,
 0.05666666666666664,
 0.05555555555555558,
 0.053333333333333344,
 0.053333333333333344,
 0.054444444444444406,
 0.05111111111111111,
 0.054444444444444406,
 0.054444444444444406,
 0.05666666666666664,
 0.05666666666666664,
 0.05555555555555558,
 0.05777777777777782,
 0.05777777777777782]
```

Double-click (or enter) to edit

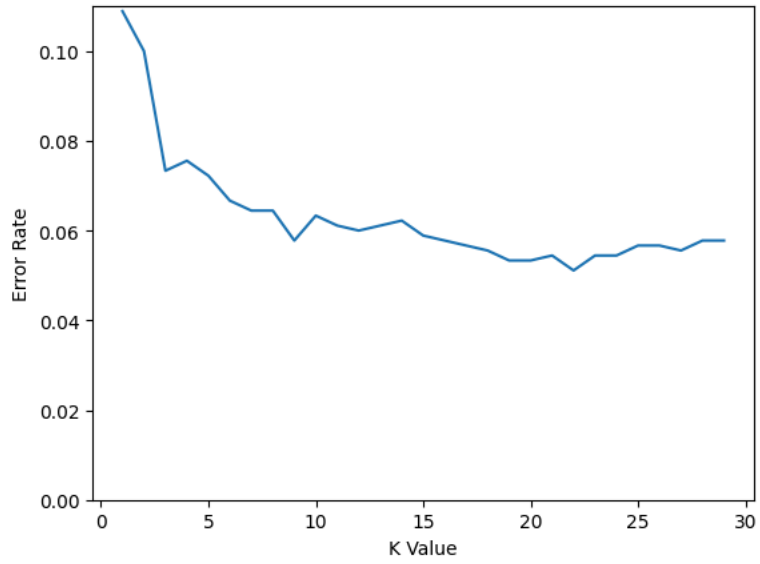
```
#plt.figure(figsize=(10,6),dpi=200)
plt.plot(range(1,30),test_error_rates,label='Test Error')
#plt.legend()
plt.ylabel('Error Rate')
plt.xlabel("K Value")
```

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




```
1 plt.plot(range(1,30),test_error_rates,label='Test Error')
2 #plt.legend()
3 plt.ylabel('Error Rate')
4 plt.xlabel("K Value")
5 plt.ylim(0, 0.11)
```

```
(0.0, 0.11)
```



Double-click (or enter) to edit

Double-click (or enter) to edit