BURSA TEKNİK ÜNİVERSİTESİ

BLM0463 Veri Madenciliğine Giriş Dersi
Proje Raporu

2022-2023 Bahar

Mehmet Emir ERDEM
20360859091

Veri Seti

- Bu projede kullanılan veri seti https://archive.ics.uci.edu/ml/datasets/Mushroom bağlantısından alınan "Mushroom" veri setidir. 8124 örneğe sahiptir. 22 öznitelik bulunmaktadır. Kayıp değerlere sahiptir. Öznitelik özellikleri kategoriktir. Sınıflandırma yapılmaktadır ve bunun için uygundur.
- Özniteliklerin detaylı bilgileri için yukarıdaki bağlantısı verilen site ziyaret edilebilir.

Uygulama

- Bu veri seti ile Python programlama dilini kullanarak, K-Nearest-Neighbor yöntemi kullanarak sınıflandırma yapılması amaçlanmıştır.
- Öncelikle gerekli kütüphaneler (pandas, numpy, seaborn, matplotlib, scikit-learn) indirilmiştir.
- Bu kütüphaneler import edilmiştir.

```
>>> import pandas as pd
>>> import numpy as np
>>> import seaborn as sns
>>> import matplotlib.pyplot as plt
>>> from sklearn.neighbors import KNeighborsClassifier
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.preprocessing import LabelEncoder
>>> from sklearn.metrics import mean_absolute_error, confusion_matrix, accuracy_score, classification_report, precis
ion_score, recall_score, f1_score
>>> from sklearn.metrics import precision_recall_curve
>>> from sklearn.metrics import roc_curve, auc
```

- Veri kümesi .csv dosya uzantısı şeklinde olup, bu haliyle gerekli kod kullanılarak eklenmiştir.

```
>>> df=pd.read_csv('C:/Users/emire/Downloads/archive/mushrooms.csv')
```

- Veri kümesinin satır ve sütun sayısı.

```
>>> df.shape
(8124, 23)
```

- Sütun Adları görüntüleme.

```
>>> df.columns
Index(['class', 'cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor',
       'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color',
       'stalk-shape', 'stalk-root', 'stalk-surface-above-ring',
       'stalk-surface-below-ring', 'stalk-color-above-ring',
       'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number',
       'ring-type', 'spore-print-color', 'population', 'habitat'],
      dtype='object')
>>>
```

- Veri türlerini gösterme.

```
>>> df.dtypes
class                     object
cap-shape                 object
cap-surface               object
cap-color                 object
bruises                   object
odor                      object
gill-attachment           object
gill-spacing              object
gill-size                 object
gill-color                object
stalk-shape               object
stalk-root                object
stalk-surface-above-ring  object
stalk-surface-below-ring  object
stalk-color-above-ring    object
stalk-color-below-ring    object
veil-type                 object
veil-color                object
ring-number               object
ring-type                 object
spore-print-color         object
population                object
habitat                   object
dtype: object
```

- Sütunlardaki eksik değerlerin olup olmadığını her veri için görüntüleme.

```
>>> df.isnull()
      class  cap-shape  cap-surface  cap-color  ...  ring-type  spore-print-color  population  habitat
0     False      False        False      False  ...      False              False       False    False
1     False      False        False      False  ...      False              False       False    False
2     False      False        False      False  ...      False              False       False    False
3     False      False        False      False  ...      False              False       False    False
4     False      False        False      False  ...      False              False       False    False
...     ...        ...          ...        ...  ...        ...                ...         ...      ...
8119  False      False        False      False  ...      False              False       False    False
8120  False      False        False      False  ...      False              False       False    False
8121  False      False        False      False  ...      False              False       False    False
8122  False      False        False      False  ...      False              False       False    False
8123  False      False        False      False  ...      False              False       False    False

[8124 rows x 23 columns]
```

- Sütunlardaki eksik değerlerin olup olmadığının toplamını görüntüleme.

```
>>> df.isnull().sum()
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: int64
>>>
```

- Verileri tablo olarak gösterme.

```
>>> df.head()
  class cap-shape cap-surface cap-color bruises  ... ring-number ring-type spore-print-color population habitat
0     p         x           s         n       t  ...           o         p                 k          s       u
1     e         x           s         y       t  ...           o         p                 n          n       g
2     e         b           s         w       t  ...           o         p                 n          n       m
3     p         x           y         w       t  ...           o         p                 k          s       u
4     e         x           s         g       f  ...           o         e                 n          a       g

[5 rows x 23 columns]
>>>
```

- Veri bilgilerini gösterme.

```
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   class                     8124 non-null   object
 1   cap-shape                 8124 non-null   object
 2   cap-surface               8124 non-null   object
 3   cap-color                 8124 non-null   object
 4   bruises                   8124 non-null   object
 5   odor                      8124 non-null   object
 6   gill-attachment           8124 non-null   object
 7   gill-spacing              8124 non-null   object
 8   gill-size                 8124 non-null   object
 9   gill-color                8124 non-null   object
 10  stalk-shape               8124 non-null   object
 11  stalk-root                8124 non-null   object
 12  stalk-surface-above-ring  8124 non-null   object
 13  stalk-surface-below-ring  8124 non-null   object
 14  stalk-color-above-ring    8124 non-null   object
 15  stalk-color-below-ring    8124 non-null   object
 16  veil-type                 8124 non-null   object
 17  veil-color                8124 non-null   object
 18  ring-number               8124 non-null   object
 19  ring-type                 8124 non-null   object
 20  spore-print-color         8124 non-null   object
 21  population                8124 non-null   object
 22  habitat                   8124 non-null   object
dtypes: object(23)
memory usage: 1.4+ MB
>>>
```

- Veri özniteliklerinin açıklanması.

```
>>> df.describe()
       class cap-shape cap-surface cap-color bruises  ... ring-number ring-type spore-print-color population habitat
count   8124      8124        8124      8124    8124  ...        8124      8124             8124       8124    8124
unique     2         6           4        10       2  ...           3         5                9          6       7
top        e         x           y         n       f  ...           o         p                w          v       d
freq    4208      3656        3244      2284    4748  ...        7488      3968             2388       4040    3148

[4 rows x 23 columns]
>>>
```

- Sütunlarda kullanılan benzersiz değerler.

```
>>> for column in df.columns:
...     unique_values = df[column].unique()
...     print("\n", f"{column}: {unique_values}")
...

 class: ['p' 'e']

 cap-shape: ['x' 'b' 's' 'f' 'k' 'c']

 cap-surface: ['s' 'y' 'f' 'g']

 cap-color: ['n' 'y' 'w' 'g' 'e' 'p' 'b' 'u' 'c' 'r']

 bruises: ['t' 'f']

 odor: ['p' 'a' 'l' 'n' 'f' 'c' 'y' 's' 'm']

 gill-attachment: ['f' 'a']

 gill-spacing: ['c' 'w']

 gill-size: ['n' 'b']

 gill-color: ['k' 'n' 'g' 'p' 'w' 'h' 'u' 'e' 'b' 'r' 'y' 'o']

 stalk-shape: ['e' 't']

 stalk-root: ['e' 'c' 'b' 'r' '?']

 stalk-surface-above-ring: ['s' 'f' 'k' 'y']

 stalk-surface-below-ring: ['s' 'f' 'y' 'k']

 stalk-color-above-ring: ['w' 'g' 'p' 'n' 'b' 'e' 'o' 'c' 'y']
```

- Tablodan sütun ve değerlerini çıkarma.

```
>>> X=df.drop('class', axis=1)
>>> y=df['class']
>>> X.head()
  cap-shape cap-surface cap-color bruises odor  ... ring-number ring-type spore-print-color population habitat
0         x           s         n       t    p  ...           o         p                 k          s       u
1         x           s         y       t    a  ...           o         p                 n          n       g
2         b           s         w       t    l  ...           o         p                 n          n       m
3         x           y         w       t    p  ...           o         p                 k          s       u
4         x           s         g       f    n  ...           o         e                 n          a       g

[5 rows x 22 columns]
>>>
```

- Çıkarılan değerleri görüntüleme.

```
>>> y.head()
0    p
1    e
2    e
3    p
4    e
Name: class, dtype: object
>>>
```

- String değerleri sayısal değerlere çevirme işlemi.

```
>>> label_encoder_x=LabelEncoder()
>>> for col in X.columns:
...     X[col]=label_encoder_x.fit_transform(X[col])
...
>>> label_encoder_y=LabelEncoder()
>>> y=label_encoder_y.fit_transform(y)
>>> X.head()
   cap-shape  cap-surface  cap-color  bruises  odor  ...  ring-number  ring-type  spore-print-color  population  habitat
0          5            2          4        1     6  ...            1          4                  2           3        5
1          5            2          9        1     0  ...            1          4                  3           2        1
2          0            2          8        1     3  ...            1          4                  3           2        3
3          5            3          8        1     6  ...            1          4                  2           3        5
4          5            2          3        0     5  ...            1          0                  3           0        1

[5 rows x 22 columns]
>>>
```

- Ayrılan ve y değişkenine atanan class değerinin sonuçları için sayısal değeri çevrilmiş hali.

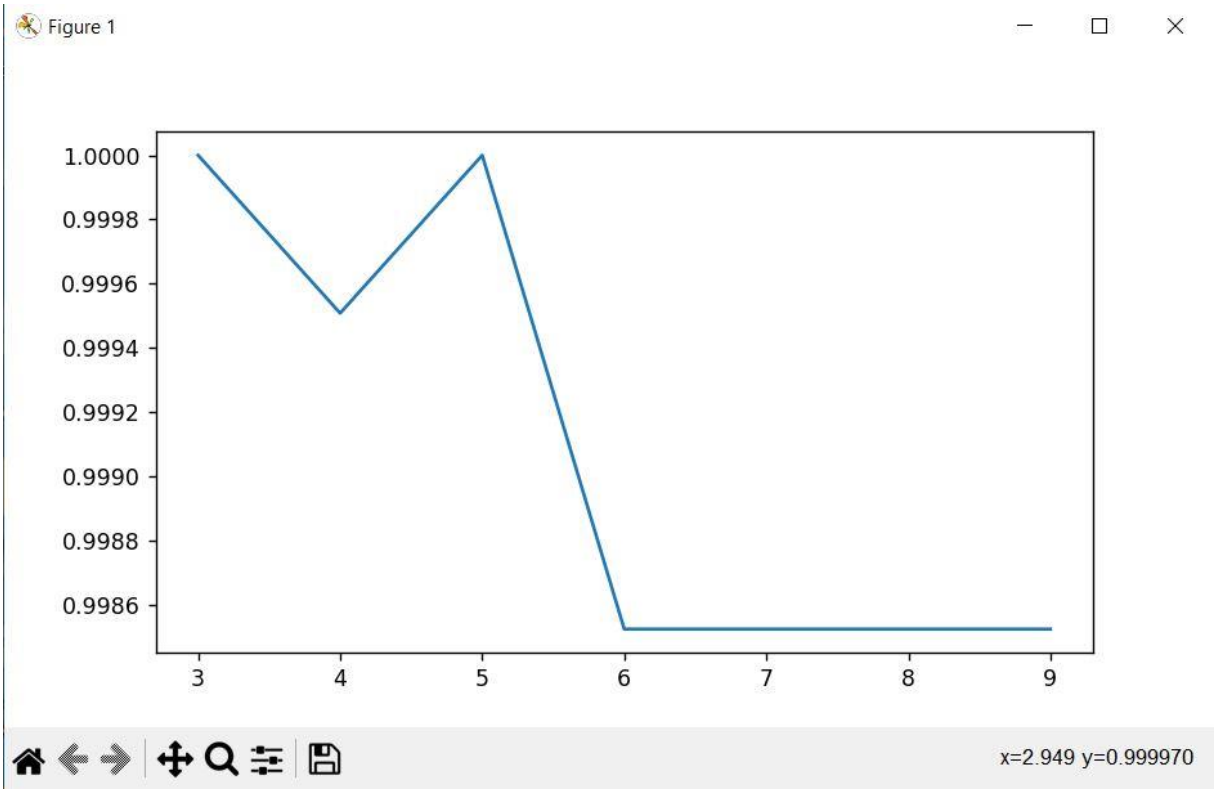
```
>>> y
array([1, 0, 0, ..., 0, 1, 0])
>>>
```

- Bu aşamadan itibaren model oluşturma kısmına geçilebilir.
- Verilerin training ve test verileri olarak bölünmesi.

```
>>> train_x,test_x,train_y,test_y=train_test_split(X,y)
>>>
```

- Minkowski uzaklık hesaplama metriği yardımıyla, K en yakın komşu algoritması yöntemi kullanılarak 3 ve 10 arasındaki en uygun k değeri belirlenmesi için her bir k değeri ile ilgili accuracy değerleri hesaplanmıştır. Bu sonuçlar grafik çizdirme fonksiyonları yardımıyla ekranda gösterilmiştir.

```
>>> for neighbors in range(3,10,1):
...     classifier = KNeighborsClassifier(n_neighbors=neighbors, metric='minkowski')
...     classifier.fit(train_x,train_y)
...     y_pred = classifier.predict(test_x)
...     acc.append(accuracy_score(test_y,y_pred))
...
KNeighborsClassifier(n_neighbors=3)
KNeighborsClassifier(n_neighbors=4)
KNeighborsClassifier()
KNeighborsClassifier(n_neighbors=6)
KNeighborsClassifier(n_neighbors=7)
KNeighborsClassifier(n_neighbors=8)
KNeighborsClassifier(n_neighbors=9)
>>> plt.figure(figsize=(15,7))
<Figure size 1500x700 with 0 Axes>
>>> plt.plot(list(range(3,10,1)), acc)
[<matplotlib.lines.Line2D object at 0x0000022CC331FE50>]
>>> plt.show()
```



- Hesaplamalar sonucunda en uygun sonucu ekrana yazdırmak için aşağıdaki kod kullanılmıştır. Bu kod sonucunda 1.0 accuracy değerini veren konum 1 (yani k=3 değeri) olarak gösterilmektedir (yukarıda görüldüğü üzere).

```
>>> print(f"Best accuracy is {np.max(acc)} and the k value is {1+acc.index(np.max(acc))}")
Best accuracy is 1.0 and the k value is 1
>>>
```

- Aşağıdaki ekran görüntüsünde görüldüğü üzere bütün k değerleri için sonuçlar denenmiş olup, hangi k değerlerinin en yüksek sonucu verdiği ve hangilerinin “mean absolute error” hata hesaplama yöntemine göre hata değerinin sonuçlandığı hesaplanmıştır. Görülen durumda en uygun kullanım değerlerinin k=3 ve k=5 değerleri olduğu görülmüştür.

```
>>> k=1+acc.index(np.max(acc))
>>> knn=KNeighborsClassifier(n_neighbors=k)
>>> knn.fit(train_x,train_y)
KNeighborsClassifier(n_neighbors=1)
>>> pred=knn.predict(test_x)
>>> print("Mean absolute error:",mean_absolute_error(pred,test_y))
Mean absolute error: 0.0
>>> k=4
>>> knn=KNeighborsClassifier(n_neighbors=k)
>>> knn.fit(train_x,train_y)
KNeighborsClassifier(n_neighbors=4)
>>> pred=knn.predict(test_x)
>>> print("Mean absolute error:",mean_absolute_error(pred,test_y))
Mean absolute error: 0.0004923682914820286
>>> k=5
>>> knn=KNeighborsClassifier(n_neighbors=k)
>>> knn.fit(train_x,train_y)
KNeighborsClassifier()
>>> pred=knn.predict(test_x)
>>> print("Mean absolute error:",mean_absolute_error(pred,test_y))
Mean absolute error: 0.0
>>> k=6
>>> knn=KNeighborsClassifier(n_neighbors=k)
>>> knn.fit(train_x,train_y)
KNeighborsClassifier(n_neighbors=6)
>>> pred=knn.predict(test_x)
>>> print("Mean absolute error:",mean_absolute_error(pred,test_y))
Mean absolute error: 0.0014771048744460858
>>> k=7
>>> knn=KNeighborsClassifier(n_neighbors=k)
>>> knn.fit(train_x,train_y)
KNeighborsClassifier(n_neighbors=7)
>>> pred=knn.predict(test_x)
>>> print("Mean absolute error:",mean_absolute_error(pred,test_y))
Mean absolute error: 0.0014771048744460858
>>> k=8
>>> knn=KNeighborsClassifier(n_neighbors=k)
>>> knn.fit(train_x,train_y)
KNeighborsClassifier(n_neighbors=8)
>>> pred=knn.predict(test_x)
>>> print("Mean absolute error:",mean_absolute_error(pred,test_y))
Mean absolute error: 0.0014771048744460858
>>> k=9
>>> knn=KNeighborsClassifier(n_neighbors=k)
>>> knn.fit(train_x,train_y)
KNeighborsClassifier(n_neighbors=9)
>>> pred=knn.predict(test_x)
>>> print("Mean absolute error:",mean_absolute_error(pred,test_y))
Mean absolute error: 0.0014771048744460858
>>>
```

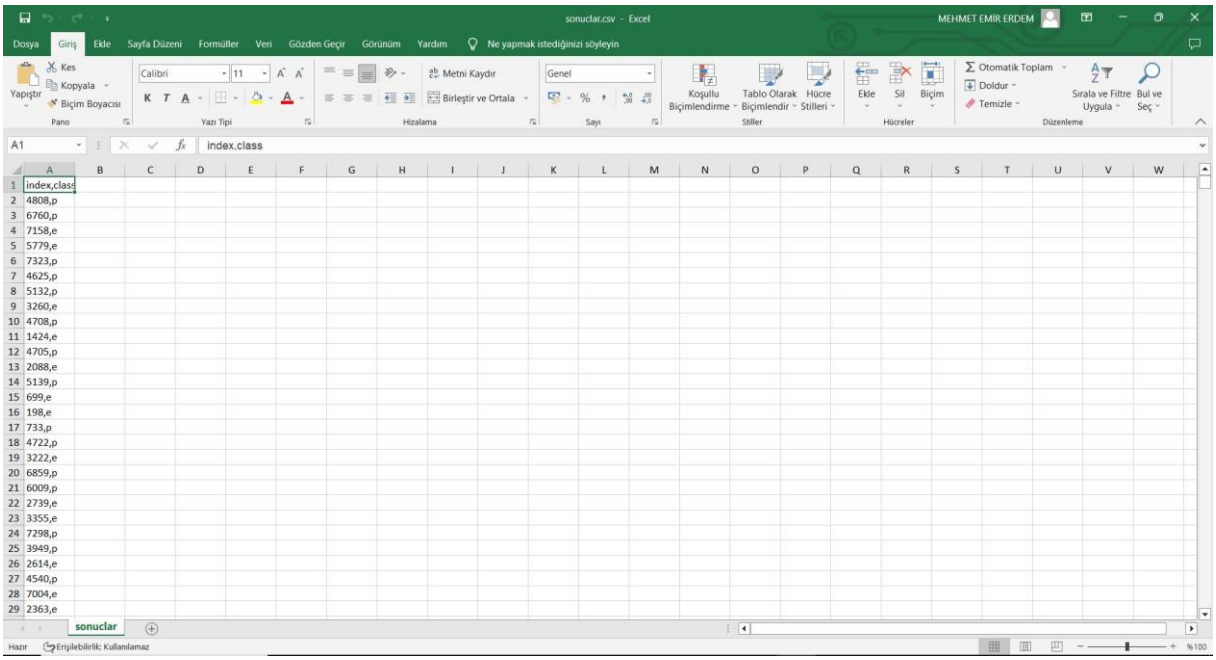
- Daha önceden ayrılmış olan x (öznitelikler) ve y (class poisonous or edible) değerleri arasında train_x , test_x ve train_y , test_y değerleri oluşturuldu. “train_x ve train_y” verileri kullanılarak hangi özniteliklere sahip verilerin zehirli veya yenilebilir olduğu KNN yöntemiyle sınıflandırılmaya çalışıldı. “y_pred” oluşturulup test_x veri setinde modelin tahmin yapması sağlandı. Çıkan sonuçlar (y_pred) ile test için ayrılmış sonuçlar (test_y) değerleri arasında accuracy hesaplaması yapıldı. Bu sonuçlar ile beraber hangi k değerinin kullanılması gerektiği sonucuna varıldı.
- Daha sonrasında en uygun k değeri ile beraber yeni bir model oluşturuldu. Bu model eğitiminde tekrardan train_x ve train_y verileri kullanılarak eğitildi. “pred” oluşturulup test_x verilerinin zehirli veya yenilebilir olduğu tahmin edildi ve bu değişkende saklandı. “mean absolute error” hata hesaplama yöntemiyle tahmin değerinin ve test_y değerinin arasındaki hata oranı hesaplandı.
- “pred2” isimli yeni bir dizi değişkeni oluşturuldu. Bu değişkende tahminlerin sonuçları tutuldu. “pred” modelinin sonuçlarına göre, eğer sayısal değerlere göre 1 döndürürse zehirli olduğu anlamına gelir ve “p” döndürür. 0 döndürürse yenilebilir olduğu anlamına gelir ve “e” döndürür. Aşağıdaki ekran görüntüsünde hangi index konumundaki mantarların zehirli veya yenilebilir olduğu sonuç olarak gösterilmiştir.

```
>>> pred2=[]
>>> for i in pred:
...     if i==1:
...             pred2.append('p')
...     else:
...             pred2.append('e')
...
>>> # pred=map(lambda x:'p' if x==1 else 'e',pred)
>>> pred2=pd.DataFrame({'index':test_x.index,'class':pred2})
>>> pred2.head()
   index class
0   4808     p
1   6760     p
2   7158     e
3   5779     e
4   7323     p
>>>
```

- Eğer sonuçlar kaydedilmek istenirse aşağıdaki kod kullanılarak kaydedilebilir.

```
>>> pred2.to_csv('sonuclar.csv', index=False)
>>>
```

- Kayıt edilmiş verilen .csv uzantılı bir dosyada görüneceği şekil aşağıdaki gibidir. Hangi index konumundaki mantarın zehirli (p) veya yenilebilir (e) olduğunu göstermektedir.



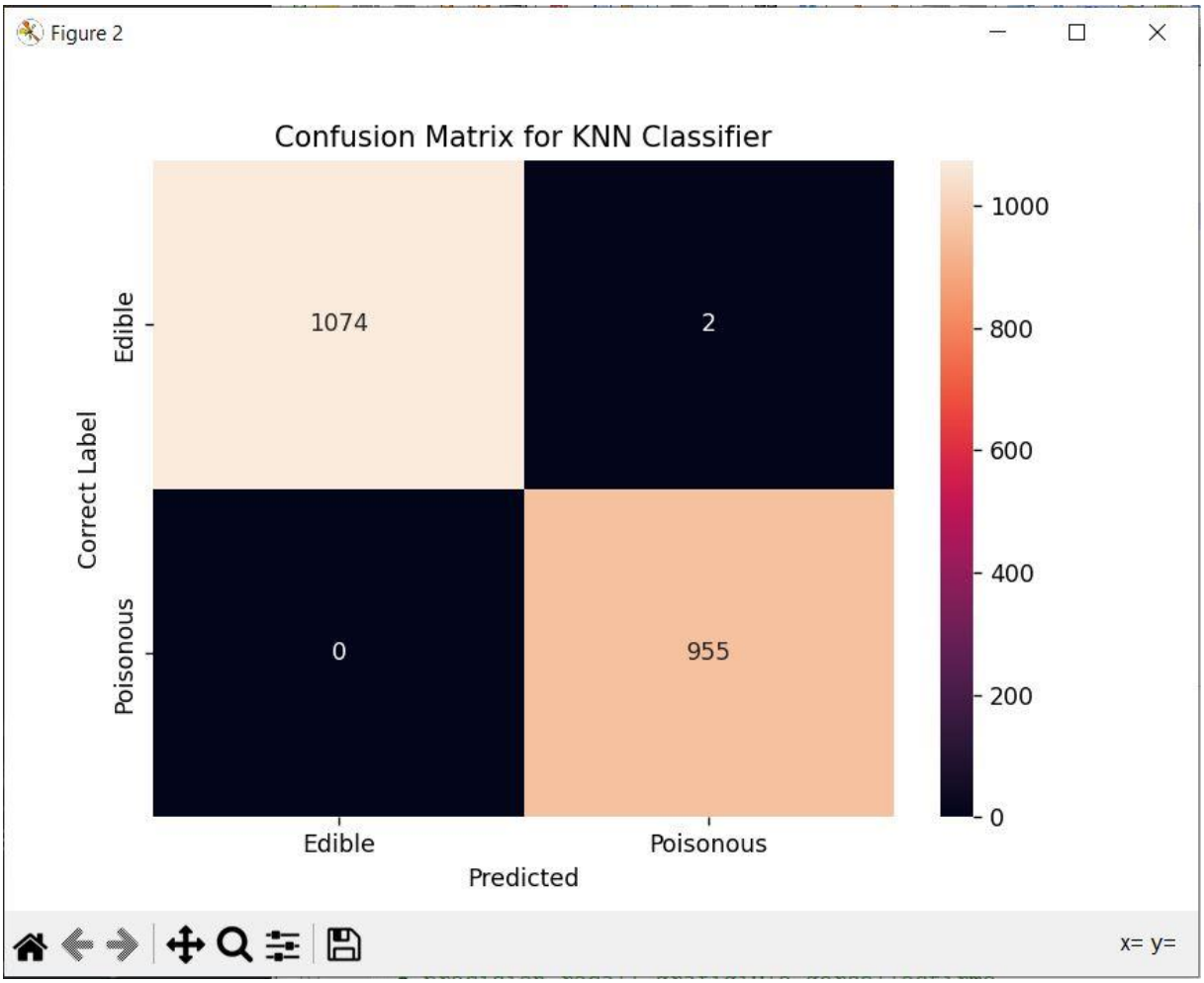
- Test verileri ile tahmin verileri arasında accuracy, precision, recall ve f1 skor değerleri hesaplanmıştır ve sonuçlar aşağıdaki ekran görüntüsünde verilmektedir.

```
>>> accuracy = accuracy_score(test_y, y_pred)
>>> precision = precision_score(test_y, y_pred)
>>> recall = recall_score(test_y, y_pred)
>>> f1 = f1_score(test_y, y_pred)
>>> report = classification_report(test_y, y_pred)
>>> print("Accuracy:", accuracy)
Accuracy: 0.999015263417036
>>> print("Precision:", precision)
Precision: 0.9979101358411703
>>> print("Recall:", recall)
Recall: 1.0
>>> print("F1-Score:", f1)
F1-Score: 0.9989539748953974
>>> print("Classification Report:", report)
Classification Report:               precision    recall  f1-score   support

           0       1.00      1.00      1.00      1076
           1       1.00      1.00      1.00       955

    accuracy                           1.00      2031
   macro avg       1.00      1.00      1.00      2031
weighted avg       1.00      1.00      1.00      2031

>>>
```

- Confusion matrix kullanılarak görüntüleme işlemi.

```
>>> KNN_confusionmatrix = confusion_matrix(test_y, y_pred)
>>> plt.figure(figsize=(7,5))
<Figure size 700x500 with 0 Axes>
>>> plt.title('Confusion Matrix for KNN Classifier')
Text(0.5, 1.0, 'Confusion Matrix for KNN Classifier')
>>> sns.heatmap(KNN_confusionmatrix , annot=True,xticklabels=["Edible", "Poisonous"], yticklabels=["Edible", "Poisonous"
],fmt='g')
<Axes: title={'center': 'Confusion Matrix for KNN Classifier'}>
>>> plt.xlabel('Predicted')
Text(0.5, 25.722222222222214, 'Predicted')
>>> plt.ylabel('Correct Label')
Text(58.222222222222214, 0.5, 'Correct Label')
>>> plt.show()
```

- Aşağıda ekran görüntüsü verilen confusion matrisinde elde bulunan sonuçlardan yenilebilir tahmin edilen doğru sonuçlardan 1074, zehirli tahmin edilen doğru sonuçlardan 955, zehirli tahmin edilen ama yenilebilir sonuçlardan 2, yenilebilir tahmin edilen ama zehirli olan sonuçlardan 0 değerleri bulunmuştur. Model neredeyse %100 doğruluğa yakın çalışmaktadır.

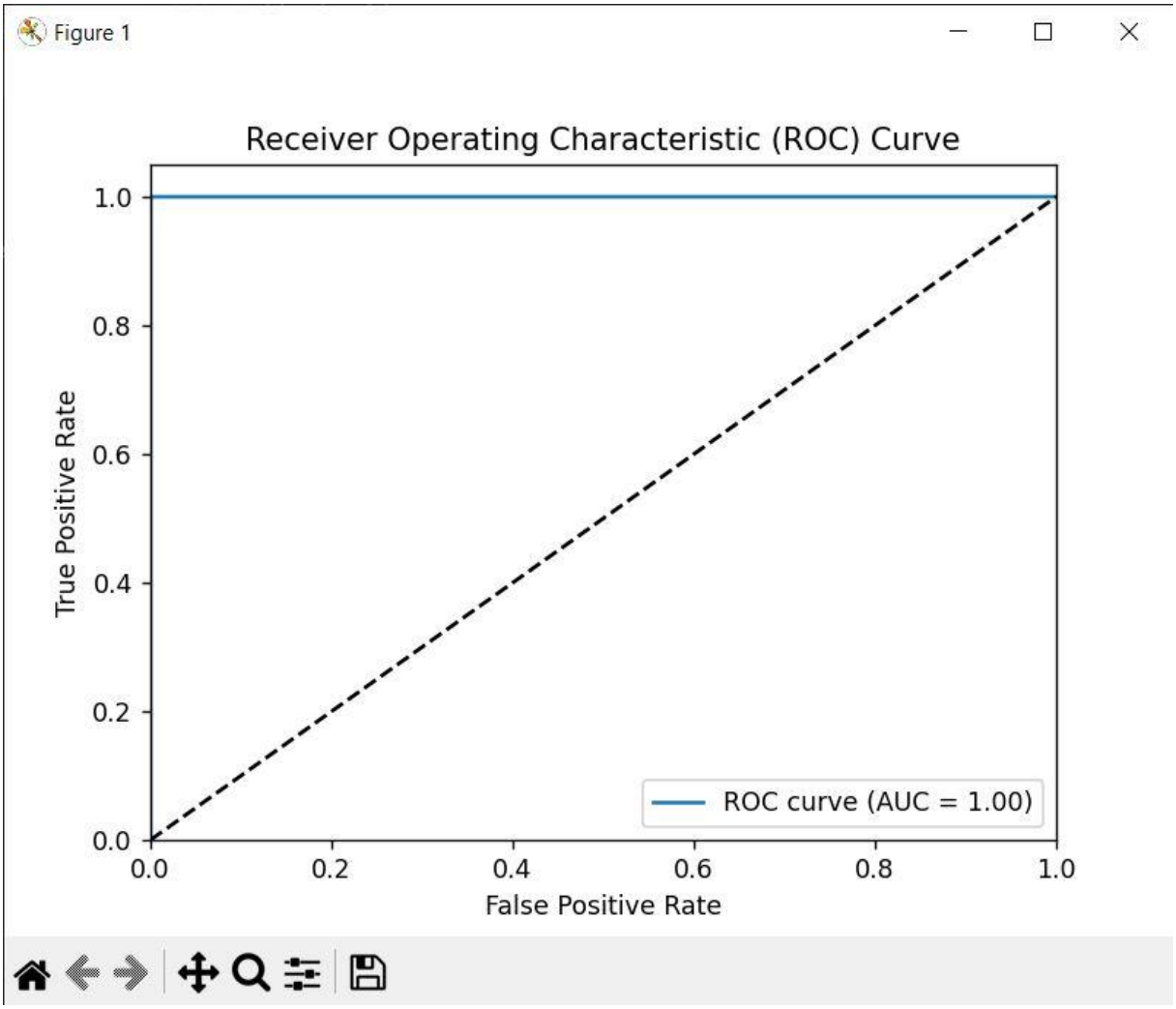


- Aynı zamanda confusion matrisinin accuracy değeri bu şekilde öğrenilebilir.

```
>>> print('Confusion matrix Accuracy is: {}'.format(metrics.accuracy_score(test_y, y_pred)))
Confusion matrix Accuracy is: 0.999015263417036
```

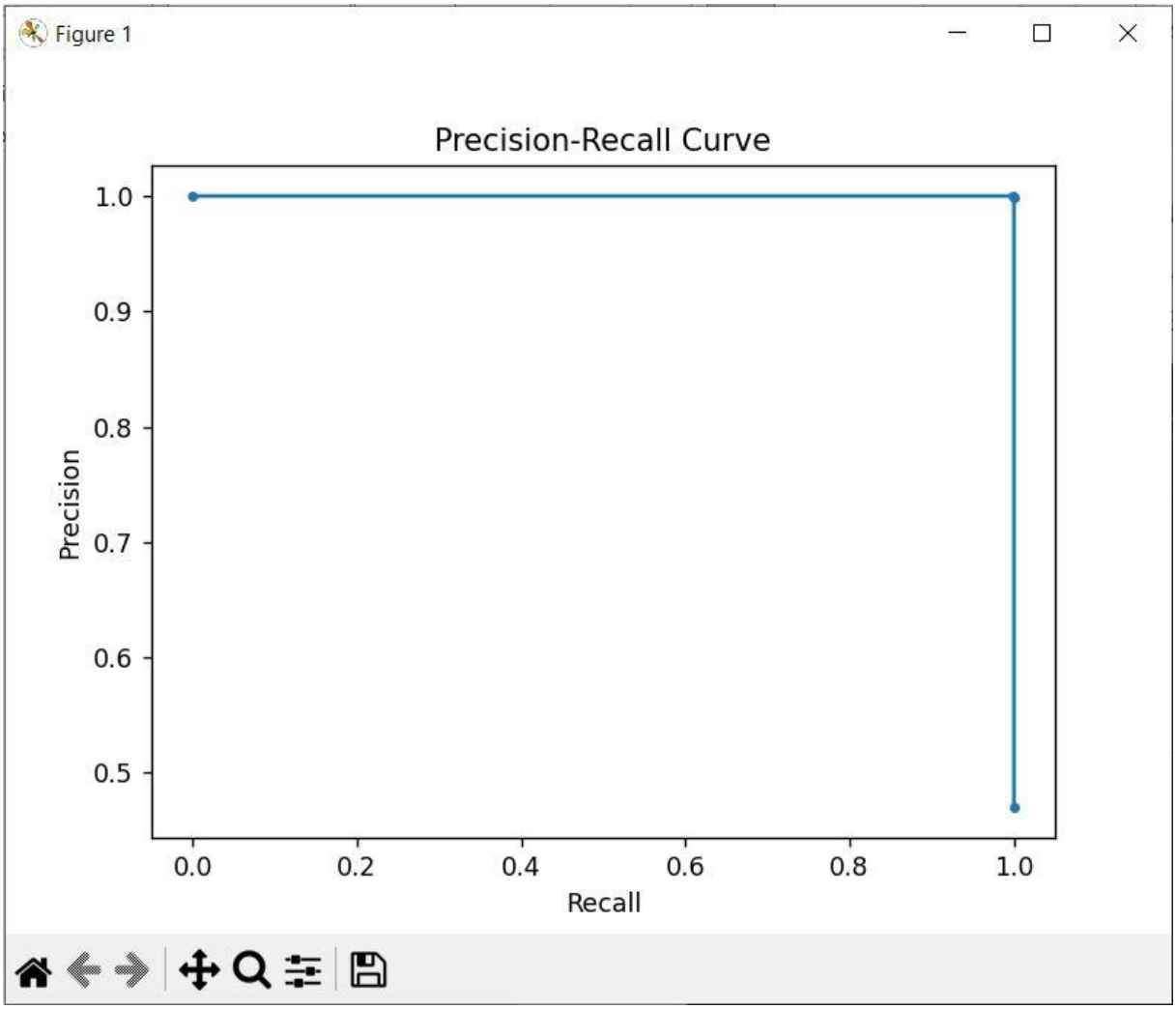
- Roc-curve grafiği oluşturma işlemi.

```
>>> y_pred_prob = knn.predict_proba(test_x)[:, 1]
>>> fpr, tpr, thresholds = roc_curve(test_y, y_pred_prob)
>>> roc_auc = auc(fpr, tpr)
>>>
>>> plt.plot(fpr, tpr, label='ROC curve (AUC = %0.2f)' % roc_auc)
[<matplotlib.lines.Line2D object at 0x0000023781EA0B50>]
>>> plt.plot([0, 1], [0, 1], 'k--')
[<matplotlib.lines.Line2D object at 0x0000023781EA0E20>]
>>> plt.xlim([0.0, 1.0])
(0.0, 1.0)
>>> plt.ylim([0.0, 1.05])
(0.0, 1.05)
>>> plt.xlabel('False Positive Rate')
Text(0.5, 0, 'False Positive Rate')
>>> plt.ylabel('True Positive Rate')
Text(0, 0.5, 'True Positive Rate')
>>> plt.title('Receiver Operating Characteristic (ROC) Curve')
Text(0.5, 1.0, 'Receiver Operating Characteristic (ROC) Curve')
>>> plt.legend(loc="lower right")
<matplotlib.legend.Legend object at 0x0000023780BBC2B0>
>>> plt.show()
```



- Precision-recall grafiği aşağıda ekran görüntüsünde verildiği gibidir. Precision değeri 1.0 bulunması sebebiyle ve recall değeri 0.99 bulunması sebebiyle grafik beklenilen şekilde oluşmuştur.

```
>>> y_pred_prob = knn.predict_proba(test_x)[:, 1]
>>> precision, recall, thresholds = precision_recall_curve(test_y, y_pred_prob)
>>> plt.plot(recall, precision, marker='.')
[<matplotlib.lines.Line2D object at 0x0000023781F20370>]
>>> plt.xlabel('Recall')
Text(0.5, 0, 'Recall')
>>> plt.ylabel('Precision')
Text(0, 0.5, 'Precision')
>>> plt.title('Precision-Recall Curve')
Text(0.5, 1.0, 'Precision-Recall Curve')
>>> plt.show()
```



Karşılaştırma

- 2021 yılında yazılan "A Comparative Study on Mushroom Classification using Supervised Machine Learning Algorithms" makalesi karşılaştırma amacıyla kullanılmıştır.
- Bu makalede veri seti %70 training %30 test verisi olarak bölünmüştür (Proje kapsamındaki çalışmada yaklaşık %75 training, %25 test olarak bölündü). Bu durum sonucunda yapılan confusion matrix işleminde aşağıdaki değerler edilmiş.

Algorithm	TP	FN	FP	TN
LR	1218	39	198	983
DT	1130	127	128	1053
KNN	1206	51	143	1038
SVM	1234	23	174	1007
NB	1218	39	211	970
RF	1206	51	139	1042

Table 2 TP, FN, FP, TN predicted by algorithms on the test set

- Bu projede kapsamında sadece KNN algoritması ile sınıflandırma yöntemi kullanılması sebebiyle KNN kısmı dikkate alınacaktır.

Algorithm	LR	DT	KNN	SVM	NB	RF
Training Accuracy	0.9061	1.0000	0.9434	0.9240	0.8983	0.9995
Average Accuracy	0.9066	0.8871	0.9291	0.9235	0.8987	0.9226
Standard Deviation	0.0103	0.0146	0.0103	0.0104	0.0113	0.0119
Test Accuracy	0.9028	0.8954	0.9204	0.9192	0.8975	0.9221

Table 3 Training accuracy, Average accuracy, Standard Deviation and Test Accuracy of algorithms

- Karşılaştırma yapılan makalenin KNN sonuçlarına bakıldığında test accuracy değerinin 0.92 olduğu görülmektedir. Bu durum makaledeki verilere göre Random Forest ile KNN yönteminin en uygun yöntem olduğunu gösterir.

- Proje kapsamında test verileri ile accuracy değeri hesaplandığında 0.99 bulunmakta. Proje kapsamında yapılan modelin daha doğru kesinlikte sonuç verdiği bu hesaplamalar sonucunda söylenebilir.

Kaynaklar

- ChatGPT
- Tank, K. (2021). A Comparative Study on Mushroom Classification using Supervised Machine Learning Algorithms.
- D. Dua and C. Graff, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.
- Veri Seti: https://archive.ics.uci.edu/ml/datasets/Mushroom
- https://www.kaggle.com/datasets/uciml/mushroom-classification