

Water Quality

Drinking water potability

Alperen Uslu

Veri Seti Hakkında

Güvenli içme suyuna erişim sağlık için esastır, temel bir insan hakkıdır ve sağlığın korunmasına yönelik etkin politikanın bir bileşenidir. Bu, ulusal, bölgesel ve yerel düzeyde bir sağlık ve kalkınma sorunu olarak önemlidir. Bazı bölgelerde, su temini ve sanitasyona yapılan yatırımların net bir ekonomik fayda sağlayabileceği gösterilmiştir, çünkü olumsuz sağlık etkileri ve sağlık bakım maliyetlerindeki azalmalar, müdahaleleri üstlenmenin maliyetlerinden daha fazladır.

water_potability.csv dosyası, 3276 farklı su kütlesi için su kalitesi ölçümlerini içerir.

Veri Seti Özellikleri Hakkında

1. pH değeri:

PH, suyun asit-baz dengesini değerlendirmede önemli bir parametredir. Aynı zamanda su durumunun asidik veya alkali durumunun göstergesidir. DSÖ, izin verilen maksimum pH sınırını 6,5 ila 8,5 arasında önermiştir. Mevcut araştırma aralıkları, DSÖ standartları aralığında olan 6,52–6,83 idi.

2. Sertlik:

Sertlik esas olarak kalsiyum ve magnezyum tuzlarından kaynaklanır. Bu tuzlar, suyun içinden geçtiği jeolojik tortulardan çözülür. Suyun sertlik oluşturan malzeme ile temas halinde olduğu süre, ham suda ne kadar sertlik olduğunu belirlemeye yardımcı olur. Sertlik başlangıçta suyun Kalsiyum ve Magnezyumun neden olduğu sabunu çökeltme kapasitesi olarak tanımlanıyordu.

Veri Seti Özellikleri Hakkında

3. Katılar (Toplam çözünmüş katılar - TDS):

Su, çok çeşitli inorganik ve bazı organik mineralleri veya potasyum, kalsiyum, sodyum, bikarbonatlar, klorürler, magnezyum, sülfatlar vb. Bu, su kullanımı için önemli bir parametredir. TDS değeri yüksek olan su, suyun yüksek oranda mineralize olduğunu gösterir. TDS için arzu edilen limit 500 mg/l ve maksimum limit 1000 mg/l olup içme amaçlı reçete edilmiştir.

4. Kloraminler:

Klor ve kloramin, kamusal su sistemlerinde kullanılan başlıca dezenfektanlardır. Kloraminler en yaygın olarak içme suyunu arıtmak için klora amonyak eklendiğinde oluşur. Litre başına 4 miligrama kadar klor seviyeleri (mg).

Veri Seti Özellikleri Hakkında

5. Sülfat:

Sülfatlar, minerallerde, toprakta ve kayalarda bulunan doğal olarak oluşan maddelerdir. Ortam havasında, yeraltı sularında, bitkilerde ve yiyeceklerde bulunurlar. Sülfatın başlıca ticari kullanımı kimya endüstrisindedir. Deniz suyundaki sülfat konsantrasyonu litre başına yaklaşık 2.700 miligramdır (mg/L). Bazı coğrafi bölgelerde çok daha yüksek konsantrasyonlar (1000 mg/L) bulunmasına rağmen, çoğu tatlı su kaynağında 3 ila 30 mg/L arasında değişir.

6. İletkenlik:

Saf su, elektrik akımını iyi bir şekilde iletmez, aksine iyi bir yalıtkandır. İyon konsantrasyonundaki artış, suyun elektriksel iletkenliğini artırır. Genel olarak, suda çözünmüş katıların miktarı elektriksel iletkenliği belirler. Elektriksel iletkenlik (EC), aslında bir çözeltinin akım iletmesini sağlayan iyonik sürecini ölçer. WHO standartlarına göre EC değeri 400 $\mu\text{S}/\text{cm}$ 'yi geçmemelidir.

Veri Seti Özellikleri Hakkında

7. Organik_karbon:

Kaynak sularındaki Toplam Organik Karbon (TOC), sentetik kaynakların yanı sıra çürüyen doğal organik maddelerden (NOM) gelir. TOC, saf sudaki organik bileşiklerdeki toplam karbon miktarının bir ölçüsüdür. US EPA'ya göre TOC olarak arıtılmış / içme suyunda < 2 mg/L ve arıtma için kullanılan kaynak suda < 4 mg/Lit.

8. Trihalometanlar:

THM'ler, klor ile işlenmiş suda bulunabilen kimyasallardır. İçme suyundaki THM'lerin konsantrasyonu, sudaki organik madde seviyesine, suyu arıtmak için gereken klor miktarına ve arıtılan suyun sıcaklığına göre değişir. 80 ppm'e kadar THM seviyeleri içme suyunda güvenli kabul edilir.

Veri Seti Özellikleri Hakkında

9. Bulanıklık:

Suyun bulanıklığı, askıda halde bulunan katı madde miktarına bağlıdır. Suyun ışık yayma özelliklerinin bir ölçüsüdür ve test koloidal maddeye göre atık deşarj kalitesini belirtmek için kullanılır. Wondo Genet Campus için elde edilen ortalama bulanıklık değeri (0,98 NTU), DSÖ tarafından tavsiye edilen 5,00 NTU değerinden düşüktür.

10. İçilebilirlik:

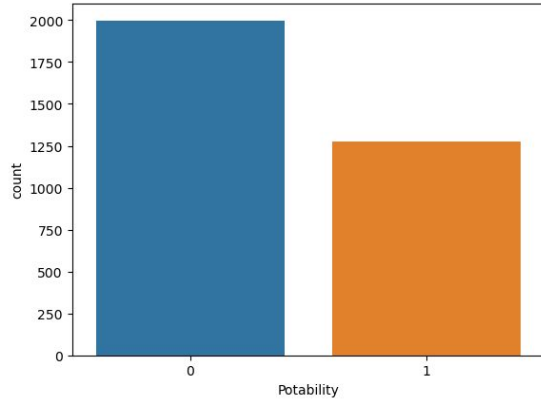
Suyun insan tüketimi için güvenli olup olmadığını gösterir; 1, İçilebilir anlamına gelir ve 0, İçilemez anlamına gelir.

Veri Kümesini Aktarma

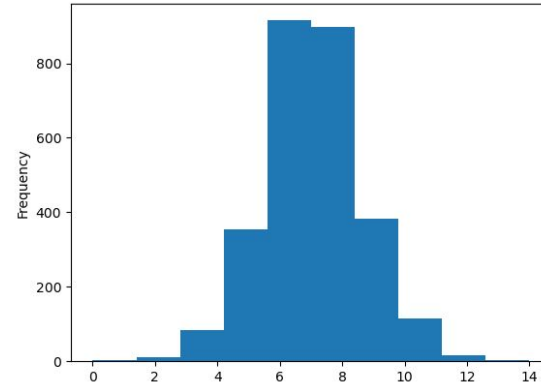
- Veri setini yüklemek için pandas kullanıyoruz.
- `import pandas as pd`
- Veri setini `df` değişkenine atayalım.
- `df=pd.read_csv("water_potability.csv")`
- Sonra veri setinin ilk 5 satırına bakalım
- `df.head()`

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|----------|------------|--------------|-------------|------------|--------------|----------------|-----------------|-----------|------------|
| 0 | NaN | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | 0 |
| 1 | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | NaN | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| 2 | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | NaN | 418.606213 | 16.868637 | 66.420093 | 3.055934 | 0 |
| 3 | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |
| 4 | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |

- Seaborn istatistiksel grafikleri çizdirmek için kullanıyoruz.
- `import seaborn as sns`
- İçilebilirlik ve içilemezlik oranına bakıyoruz.
- `sns.countplot(x="Potability", data=df)`
-



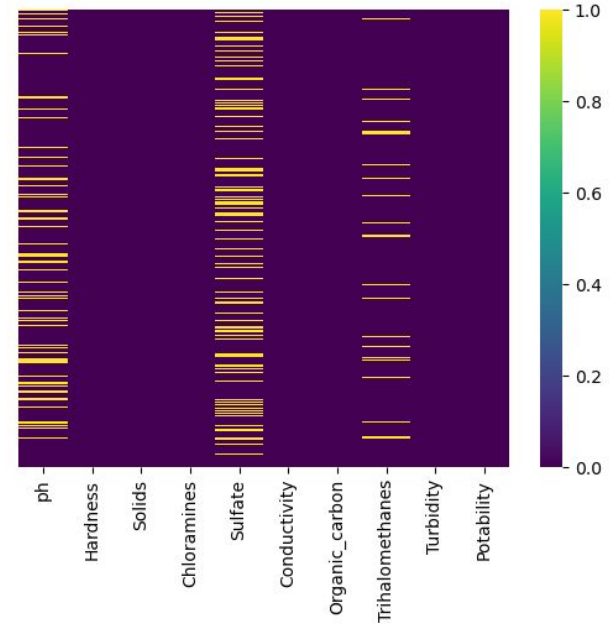
- ph sıklığına bakmak içinde
- `df["ph"].plot.hist()`
-



- Boş değer olup olmadığına kontrol ediyoruz.
- `df.isnull().sum()`

```
ph          491
Hardness      0
Solids        0
Chloramines   0
Sulfate       781
Conductivity  0
Organic_carbon 0
Trihalomethanes 162
Turbidity     0
Potability    0
dtype: int64
```

- Isı haritasıyla nerelerde null değerler olduğuna bakıyoruz.
- `sns.heatmap(df.isnull(),yticklabels=False,cmap="viridis")`
-



- Null olan değerlere ortalama atadık
- `df.fillna(df.mean(),inplace=True)`

- Aykırı değerleri bulmak için
- `from sklearn.neighbors import LocalOutlierFactor` kullanıyoruz
- Bazı kodlardan sonra sonuç aşağıdaki gibi aykırı değerler çıkıyor

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|----------|------------|--------------|-------------|------------|--------------|----------------|-----------------|-----------|------------|
| 0 | 7.080795 | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | 0 |
| 1 | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | 333.775777 | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| 2 | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | 333.775777 | 418.606213 | 16.868637 | 66.420093 | 3.055934 | 0 |
| 3 | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |
| 4 | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|------|---------|------------|--------------|-------------|------------|--------------|----------------|-----------------|-----------|------------|
| 1031 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 1068 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 1186 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 1537 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 1554 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 2134 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 2602 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 2704 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 2718 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 2737 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 2906 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 3149 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 3150 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |
| 3161 | 6.60254 | 174.632977 | 21607.483238 | 6.581327 | 308.931421 | 657.570422 | 9.064445 | 68.827047 | 3.592496 | 1 |

- Son durumda verimiz aşağıdaki gibidir.

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|----------|------------|--------------|-------------|------------|--------------|----------------|-----------------|-----------|------------|
| 0 | 7.080795 | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | 0 |
| 1 | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | 333.775777 | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| 2 | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | 333.775777 | 418.606213 | 16.868637 | 66.420093 | 3.055934 | 0 |
| 3 | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |
| 4 | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |

- Veri seti hakkında bilgiler
- `df.info()` bakıyoruz
- `df.info()`

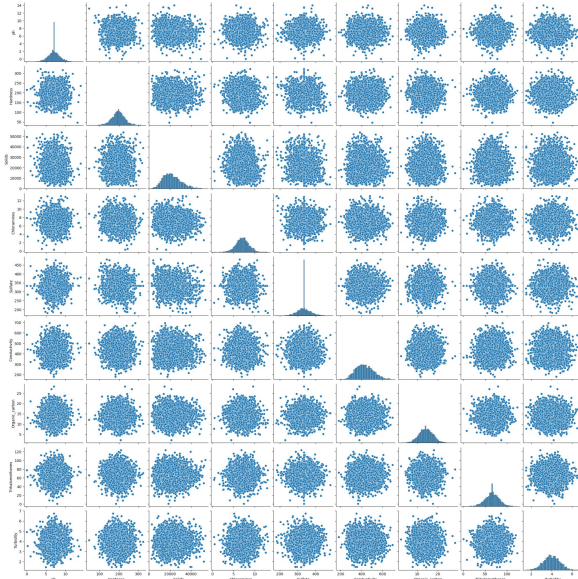
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ph                     3276 non-null   float64
1   Hardness               3276 non-null   float64
2   Solids                 3276 non-null   float64
3   Chloramines            3276 non-null   float64
4   Sulfate                3276 non-null   float64
5   Conductivity           3276 non-null   float64
6   Organic_carbon         3276 non-null   float64
7   Trihalomethanes        3276 non-null   float64
8   Turbidity              3276 non-null   float64
9   Potability             3276 non-null   int64
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

- Veri setinin ortalaması, standart sapması hesaplamak için
- `df.describe().T` kullanıyoruz.

| | count | mean | std | min | 25% | 50% | 75% | max |
|-----------------|--------|--------------|-------------|-------------|--------------|--------------|--------------|--------------|
| ph | 3276.0 | 7.080345 | 1.466368 | 0.000000 | 6.283265 | 7.080795 | 7.865260 | 14.000000 |
| Hardness | 3276.0 | 196.284533 | 32.803464 | 47.432000 | 176.724139 | 196.778920 | 216.576645 | 323.124000 |
| Solids | 3276.0 | 21981.372303 | 8599.331395 | 2552.962804 | 15715.740730 | 20988.258608 | 27286.909002 | 53735.899194 |
| Chloramines | 3276.0 | 7.122298 | 1.576932 | 0.352000 | 6.133588 | 7.121371 | 8.110201 | 13.127000 |
| Sulfate | 3276.0 | 333.716944 | 35.813602 | 180.206746 | 316.775351 | 333.775777 | 350.207431 | 481.030642 |
| Conductivity | 3276.0 | 426.820402 | 81.631990 | 181.483754 | 365.842780 | 422.033283 | 482.303317 | 695.369528 |
| Organic_carbon | 3276.0 | 14.261592 | 3.319246 | 2.200000 | 12.033868 | 14.201611 | 16.542410 | 28.300000 |
| Trihalomethanes | 3276.0 | 66.398835 | 15.743965 | 0.738000 | 56.714388 | 66.396293 | 76.632089 | 124.000000 |
| Turbidity | 3276.0 | 3.966010 | 0.779624 | 1.450000 | 3.440832 | 3.950917 | 4.499166 | 6.739000 |
| Potability | 3276.0 | 0.391941 | 0.488258 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 |

- İçilebilirlik oranına bakmak için
- `df.Potability.value_counts()` kullanıyoruz
- 0 1992
- 1 1284
- X girdisini ve y çıktısını oluşturuyoruz
- `X=df.drop("Potability",axis=1)`
- `y=df["Potability"]`

- Daha sonra girdi verisindeki sayısal değişkenlerin ikili saçılım grafiklerini çizdirelim.
- `sns.pairplot(X);`



- Korelasyona bakmak için
- `corr = df.corr()`
- `print(corr)` kullanıyoruz

| | ph | Hardness | Solids | Chloramines | Sulfate | \ |
|-----------------|-----------|-----------|-----------|-------------|-----------|---|
| ph | 1.000000 | 0.075947 | -0.084071 | -0.028882 | 0.019745 | |
| Hardness | 0.075947 | 1.000000 | -0.041262 | -0.028854 | -0.094008 | |
| Solids | -0.084071 | -0.041262 | 1.000000 | -0.063214 | -0.139035 | |
| Chloramines | -0.028882 | -0.028854 | -0.063214 | 1.000000 | 0.015921 | |
| Sulfate | 0.019745 | -0.094008 | -0.139035 | 0.015921 | 1.000000 | |
| Conductivity | 0.014047 | -0.032427 | 0.011755 | -0.024299 | -0.017366 | |
| Organic_carbon | 0.043010 | 0.008088 | 0.012807 | -0.011745 | 0.028066 | |
| Trihalomethanes | 0.003790 | -0.015453 | -0.007261 | 0.018298 | -0.026874 | |
| Turbidity | -0.037039 | -0.013786 | 0.017439 | 0.005088 | -0.006122 | |
| Potability | -0.005718 | -0.017380 | 0.035616 | 0.022983 | -0.023541 | |

| | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | \ |
|-----------------|--------------|----------------|-----------------|-----------|---|
| ph | 0.014047 | 0.043010 | 0.003790 | -0.037039 | |
| Hardness | -0.032427 | 0.008088 | -0.015453 | -0.013786 | |
| Solids | 0.011755 | 0.012807 | -0.007261 | 0.017439 | |
| Chloramines | -0.024299 | -0.011745 | 0.018298 | 0.005088 | |
| Sulfate | -0.017366 | 0.028066 | -0.026874 | -0.006122 | |
| Conductivity | 1.000000 | 0.002691 | 0.004118 | 0.002434 | |
| Organic_carbon | 0.002691 | 1.000000 | -0.013972 | -0.022177 | |
| Trihalomethanes | 0.004118 | -0.013972 | 1.000000 | -0.021934 | |
| Turbidity | 0.002434 | -0.022177 | -0.021934 | 1.000000 | |
| Potability | 0.006114 | -0.035499 | 0.008273 | -0.002138 | |

| | Potability |
|-----------------|------------|
| ph | -0.005718 |
| Hardness | -0.017380 |
| Solids | 0.035616 |
| Chloramines | 0.022983 |
| Sulfate | -0.023541 |
| Conductivity | 0.006114 |
| Organic_carbon | -0.035499 |
| Trihalomethanes | 0.008273 |
| Turbidity | -0.002138 |
| Potability | 1.000000 |

- Test ve Train oluşturmak için sklearn kütüphanesinden çağırıyoruz
- `from sklearn.model_selection import train_test_split`
- `X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=100)`
- `print(X_train.shape)`
- `print(X_test.shape)`
- `print(y_train.shape)`
- `print(y_train.shape)`

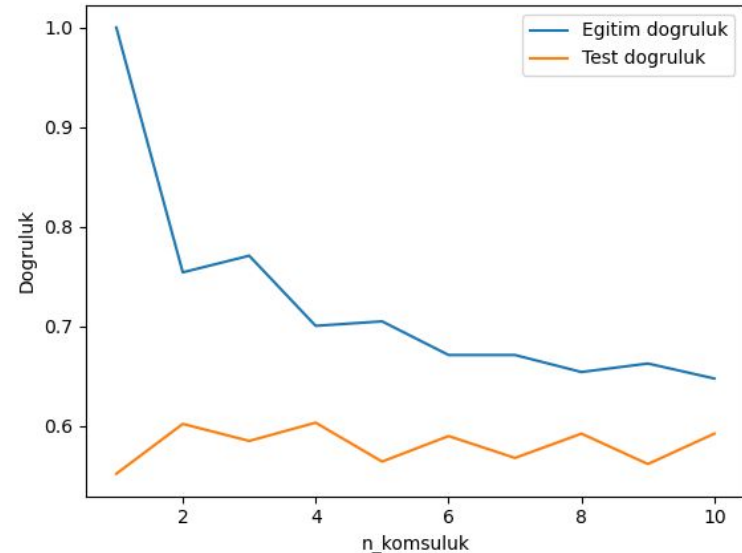
- Logistic Regression
- `from sklearn.linear_model import LogisticRegression` çağırıyoruz
- `lg_model=LogisticRegression()`
- `lg_model.fit(X_train,y_train)`
- `lg_model.score(X_test,y_test)`
- 0.6056166056166056
- `lg_model.score(X_train,y_train)`
- 0.6084656084656085

- Confusion Matrix
- `from sklearn.metrics import confusion_matrix`
- `tahmin=lg_model.predict(X_test)`
- `confusion_matrix(y_test,tahmin)`
- `confusion_matrix(y_test,tahmin)`
- `array([[496, 1],`
- `[322, 0]],`
-

- Naive Bayes
- `from sklearn.naive_bayes import GaussianNB` kullanıyoruz
- `model=GaussianNB()`
- `nb_b=model.fit(X_train,y_train)`
- `nb_b.predict(X_test)[0:10]` on tahmin yapar
- `tahmin=nb_b.predict(X_test)`
- `confusion_matrix(y_test,tahmin)`
- `array([[445, 52],`
- `[252, 70]],)`
- `y_pred=nb_b.predict(X_test)`
- `from sklearn.metrics import accuracy_score` ile sonuçları ölçüyoruz
- `accuracy_score(y_test,y_pred)`
- 0.6288156288156288

- KNN en yakın komşu uzaklığı hesaplama
- `from sklearn.neighbors import`
`KNeighborsClassifier`
- `knn=KNeighborsClassifier()`
- `knn_model=knn.fit(X_train,y_train)` `knn_model`
- `y_pred=knn_model.predict(X_test)`
- accuracy score'nu hesaplıyoruz
- `accuracy_score(y_test,y_pred)`
- 0.5641025641025641
- 3 en yakın komşuluk sayısına göre
- `reg=KNeighborsClassifier(n_neighbors=3)`
- `reg.fit(X_train,y_train)`
- Doğruluk oranı
- `reg.score(X_test,y_test)`
- 0.5427435387673957

- `import matplotlib.pyplot as plt`
- Matplotlib kullanarak KNN'in grafiğini çıkarıyoruz



- Karar Ağaçları oluşturma
- `from sklearn.tree import DecisionTreeClassifier`
- `tree = DecisionTreeClassifier()`
- `tree.fit(X_train,y_train)`
- Eğitim ve test verisi doğruluğu
- `print(tree.score(X_train,y_train))`
- `print(tree.score(X_test,y_test))`
- 1.0
- 0.6123260437375746

- Kmeans Algoritması
- Benzer özellikleri kümeler import ediyoruz
- `from sklearn.cluster import KMeans`
- `kmeans=KMeans(n_clusters=4)`
- `kmeans.fit(X)`
- verileri tahmin ediyoruz
- `labels = kmeans.predict(X)`
- labels
- `array([0, 0, 2, ..., 3, 2, 1])`
-

- Random Forest farklı karar ağaçları için
- `from sklearn.ensemble import`
`RandomForestClassifier`
- `rf =`
`RandomForestClassifier(n_estimators=100,`
`random_state=42)`
- `rf.fit(X_train, y_train)`
- `y_pred = rf.predict(X_test)`
- `accuracy = rf.score(X_test, y_test)`
- `print("Accuracy:", accuracy)`
- 0.6495726495726496

- SVC veri setinde verileri iki sınıfa ayırmak için
- `from sklearn.svm import SVC`
- `svc = SVC(kernel='linear')`
- `svc.fit(X_train, y_train)`
- `y_pred = svc.predict(X_test)`
- `from sklearn.metrics import`
`accuracy_score`
- `acc = accuracy_score(y_test, y_pred)`
- `print("Accuracy: ", acc)`
- Accuracy: 0.608058608058608

- En iyi KNN bulabilmek için
- `from sklearn.neighbors import`
`KNeighborsClassifier`
- `from sklearn.model_selection import`
`GridSearchCV`
- `param_grid = {'n_neighbors':[3,5,7,9],`
`'metric': ['euclidean', 'manhattan']}`
- `knn = KNeighborsClassifier()`
- `grid_search = GridSearchCV(knn,`
`param_grid, cv=5)`
- `grid_search.fit(X_train, y_train)`

- `print("En iyi hiperparametreler: ",`
`grid_search.best_params_)`
- `best_train_score =`
`grid_search.score(X_train, y_train)`
- `best_test_score =`
`grid_search.score(X_test, y_test)`
- `print("Eğitim skoru:", best_train_score)`
- `print("Test skoru:", best_test_score)`
- En iyi hiperparametreler: {'metric':
'manhattan', 'n_neighbors': 9}
- Eğitim skoru: 0.6621896621896622
- Test skoru: 0.5787545787545788

Karşılaştırılmaları

- lr = LogisticRegression()
- nb = GaussianNB()
- dt = DecisionTreeClassifier()
- rf = RandomForestClassifier()
- svm = SVC()
- knn = KNeighborsClassifier()
- # Algoritmaları eğitme
- lr.fit(X_train, y_train)
- nb.fit(X_train, y_train)
- dt.fit(X_train, y_train)
- rf.fit(X_train, y_train)
- svm.fit(X_train, y_train)
- knn.fit(X_train, y_train)
- # Performanslarını karşılaştırma
- y_pred_lr = lr.predict(X_test)
- y_pred_nb = nb.predict(X_test)
- y_pred_dt = dt.predict(X_test)
- y_pred_rf = rf.predict(X_test)
- y_pred_svm = svm.predict(X_test)
- y_pred_knn = knn.predict(X_test)

- acc_lr = accuracy_score(y_test, y_pred_lr)
- acc_nb = accuracy_score(y_test, y_pred_nb)
- acc_dt = accuracy_score(y_test, y_pred_dt)
- acc_rf = accuracy_score(y_test, y_pred_rf)
- acc_svm = accuracy_score(y_test, y_pred_svm)
- acc_knn = accuracy_score(y_test, y_pred_knn)
-
- print("Logistic Regression accuracy: ", acc_lr)
- print("Naive Bayes accuracy: ", acc_nb)
- print("Decision Tree accuracy: ", acc_dt)
- print("Random Forest accuracy: ", acc_rf)
- print("Support Vector Machine accuracy: ", acc_svm)
- print("K-Nearest Neighbors accuracy: ", acc_knn)
-
- Logistic Regression accuracy: 0.6056166056166056
- Naive Bayes accuracy: 0.6288156288156288
- Decision Tree accuracy: 0.5714285714285714
- Random Forest accuracy: 0.6556776556776557
- Support Vector Machine accuracy: 0.6068376068376068
- K-Nearest Neighbors accuracy: 0.5641025641025641
-

Yapay Sinir Ağları

- Oluşturmak için tensorflow import ediyoruz
- `import tensorflow as tf`
- Daha sonra çıktı eğitim ve çıktı test verilerini one-hot kodlayalım.
- `y_train=tf.keras.utils.to_categorical(y_train)`
`y_test=tf.keras.utils.to_categorical(y_test)`
- ilk beş satır
- `y_train[:5,:]`
- `array([[1., 0.], [0., 1.], [0., 1.], [1., 0.], [0., 1.]`,
`dtype=float32)`
- numpy array yapısına çeviriyoruz
-
- `X_train=X_train.values`
`X_test=X_test.values`
- Sequential sınıfını import edelim.
- `from tensorflow.keras.models import Sequential`
- Dense sınıfını import edelim.
- `from tensorflow.keras.layers import Dense`
- `model=Sequential()`

- 2 kategori olduğu için bu layerdaki nöron sayısı 2 ve aktivasyon fonksiyonu olarak sigmoid.
- ```
model.add(Dense(64,activation="relu",
 input_shape=X_train[0].shape))
model.add(Dense(128,activation="relu"))
model.add(Dense(128,activation="relu"))
model.add(Dense(128,activation="relu"))
model.add(Dense(64,activation="relu"))
model.add(Dense(64,activation="relu"))
model.add(Dense(64,activation="relu"))
model.add(Dense(64,activation="relu"))
model.add(Dense(2,activation="sigmoid"))
```

- ne kadar iyi tahmin yaptığı
- ```
model.compile(optimizer="adam",
              loss="categorical_crossentropy",
              metrics=["acc"])
```
- ```
history=model.fit(X_train,y_train,
 batch_size=32,
 epochs=7,
 validation_split=0.1)
```

```
Epoch 1/7
70/70 [=====] - 1s 9ms/step - loss: 8.5150 - acc: 0.5075 - val_loss: 9.3521 - val_acc: 0.4268
Epoch 2/7
70/70 [=====] - 0s 4ms/step - loss: 3.8137 - acc: 0.4966 - val_loss: 4.3418 - val_acc: 0.4268
Epoch 3/7
70/70 [=====] - 0s 4ms/step - loss: 2.6346 - acc: 0.5138 - val_loss: 1.2779 - val_acc: 0.5732
Epoch 4/7
70/70 [=====] - 0s 4ms/step - loss: 1.0462 - acc: 0.5328 - val_loss: 0.6872 - val_acc: 0.5732
Epoch 5/7
70/70 [=====] - 0s 5ms/step - loss: 0.7566 - acc: 0.5540 - val_loss: 0.7212 - val_acc: 0.5732
Epoch 6/7
70/70 [=====] - 0s 5ms/step - loss: 0.7546 - acc: 0.5563 - val_loss: 0.7392 - val_acc: 0.5732
Epoch 7/7
70/70 [=====] - 0s 4ms/step - loss: 0.7250 - acc: 0.5568 - val_loss: 0.7941 - val_acc: 0.4309
```

- ```
model.evaluate(X_test,y_test)
```

```
26/26 [=====] - 0s 2ms/step - loss: 0.8010 - acc: 0.3956
[0.8010177612304688, 0.3956044018268585]
```

Bitti

kaynak:<https://www.kaggle.com/datasets/adityakadiwal/water-potability>
