Preprocessing_ML_Days

June 21, 2020

1 Preprocessing

```
In [2]: import pandas as pd
        import numpy as np

        dataset = {"sim": ["Sevdanur", "Selcuk", "Huseyin", "Dogus", "Haticenur", "Meltem"],
                   "Soyad": ["Genc", "Genc", "Sahin", "Can", "Nalbant", "Onder"],
                   "Yas": [24, 22, 24, 23, "bilinmiyor", 23],
                   "Sehir": ["Bursa", "Ankara", "Istanbul", np.nan, "Izmir", "Istanbul"],
                   "Ulke": ["Turkiye", "Turkiye", "Turkiye", "Turkiye", "Turkiye", "Turkiye"],
                   "GANO": [np.nan, np.nan, np.nan, np.nan, 3.90, np.nan]}

        df = pd.DataFrame(dataset)
        df
```

Out[2]:

	sim	Soyad	Yas	Sehir	Ulke	GANO
0	Sevdanur	Genc	24	Bursa	Turkiye	NaN
1	Selcuk	Genc	22	Ankara	Turkiye	NaN
2	Huseyin	Sahin	24	Istanbul	Turkiye	NaN
3	Dogus	Can	23	NaN	Turkiye	NaN
4	Haticenur	Nalbant	bilinmiyor	Izmir	Turkiye	3.9
5	Meltem	Onder	23	Istanbul	Turkiye	NaN

---

2 1. Adm: Büyük resime bakn!

Her eyden önce, bir preprocessing ilemine balarken, veri tiplerine, satr-sütün saylarna, eksik verilere ve genel emaya bakarak balamalsnz. Burada `<DataFrame>.info()` fonksiyonu ile bir önbilgi alnabilir.

- lk dikkatimi çeken unsur `Yas` kolonunun integer olmas yerine object olmas. Dataframe'e dönüp baktmda yalardan birinin bilinmiyor olarak kodlandn görüyorum. Eer saylardan oluan bir kolonda farkl bir datatype varsa, pandas bunun object olarak alglayacaktr.
- Dikkatimi çeken dier bir unsur `Sehir` ve `GANO` kolonundaki eksik deerler, bunlarn halledilmesi gerekecek.

- Toplam 6 satr olmasna ramen `GANO` kolonunda sadece tek bir deer görebiliyorum, burada bu kolonu tamamen kaldrmak mantkl olacan düünüyorum.
- Ulke kolonundaki tüm deerler ayn, bu yüzden kaldrabiliriz.

```
In [4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
sim     6 non-null object
Soyad    6 non-null object
Yas      6 non-null object
Sehir    5 non-null object
Ulke     6 non-null object
GANO     1 non-null float64
dtypes: float64(1), object(5)
memory usage: 368.0+ bytes
```

2.1 NaN kontrolü

NaN deerleri saydrarak kontrol edelim. Eer bir kez `.sum()` fonksiyonunu çarrsam, kolon baznda toplayacaktr, eer bir kez daha çarrsam, eksik deerlerimin toplamn da görebilirim.

```
In [6]: df.isna()
```

```
Out[6]:     sim  Soyad    Yas  Sehir   Ulke   GANO
        0  False  False  False  False  False   True
        1  False  False  False  False  False   True
        2  False  False  False  False  False   True
        3  False  False  False   True  False   True
        4  False  False  False  False  False  False
        5  False  False  False  False  False   True
```

```
In [7]: df.isna().sum()
```

```
Out[7]: sim     0
        Soyad    0
        Yas      0
        Sehir    1
        Ulke     0
        GANO     5
        dtype: int64
```

```
In [8]: df.isna().sum().sum()
```

```
Out[8]: 6
```

3 2. Adm: Manipülasyona Balayn!

3.1 Bilgi içermeyen kolonlarn kaldrlmas

GANO ve Ulke satrlarnn kaldrlmasna karar vermitik, bunu yapabileceimiz iki yöntem var: - Önkabul olarak, eer kolonlar belirli bir eik deerinin üzerinde *NaN* deer içerdiinde kaldrmak istiyorsanz - Seçtiiniz kolonlar manuel olarak kaldrmak istiyorsanz

```
In [13]: # 1. Yöntem
         df.dropna(axis=1, how="any", thresh=3) # GANO Sütununun kaldrlmas

         # axis : coloumn'da mi yoksa row'da mi yapilacak o belirleniyor.
         # how : any / all herhangi bir null iceriyorsa onu getiriyor.
         # thresh : nan sayisi bu deger uzerinde cikarsa goster (thershold)
         # bu kod ile kolonlar arasinda 3 taneden daha fazla nan degerler varsa onlari dusurme
         # GANO sutununda 3'ten fazla nan oldugu icin artik o sutunu listelerken getirmiyor.
```

Out[13]:

	sim	Soyad	Yas	Sehir	Ulke
0	Sevdanur	Genc	24	Bursa	Turkiye
1	Selcuk	Genc	22	Ankara	Turkiye
2	Huseyin	Sahin	24	Istanbul	Turkiye
3	Dogus	Can	23	NaN	Turkiye
4	Haticenur	Nalbant	bilinmiyor	Izmir	Turkiye
5	Meltem	Onder	23	Istanbul	Turkiye

```
In [14]: # 2. Yöntem
         df.drop(labels=["GANO"], axis=1)

         # Eer ayn dataframe'inize direk uygulamak istiyoranz inplace parametresine True deeri
         # df.drop(labels=["GANO"], axis=1, inplace=True)

         # ozellikle bir column u silmek istersek drop ile yapabiliriz.
```

Out[14]:

	sim	Soyad	Yas	Sehir	Ulke
0	Sevdanur	Genc	24	Bursa	Turkiye
1	Selcuk	Genc	22	Ankara	Turkiye
2	Huseyin	Sahin	24	Istanbul	Turkiye
3	Dogus	Can	23	NaN	Turkiye
4	Haticenur	Nalbant	bilinmiyor	Izmir	Turkiye
5	Meltem	Onder	23	Istanbul	Turkiye

Ayrca unutmadan Ulke satrndaki her deer ayn olduu için modelimizin buna ihtiyac olmayacak.

```
In [15]: df.drop(labels=["GANO", "Ulke"], axis=1, inplace=True)

         # inplace : bunu dataframe e uygula, direk uzerine yaz! orjinali silinir.

In [16]: df
```

```
Out[16]:         sim    Soyad         Yas     Sehir
          0   Sevdanur     Genc          24     Bursa
          1     Selcuk     Genc          22    Ankara
          2    Huseyin    Sahin          24  Istanbul
          3      Dogus      Can          23       NaN
          4  Haticenur  Nalbant  bilinmiyor     Izmir
          5     Meltem    Onder          23  Istanbul
```

3.2 Eksik deerlerin halledilmesi

Eksik deerlerin halledilmesiyle ilgili basit ve daha kompleks yöntemler var, burada amaç verisetimizde dezenformasyon yaratmadan bu problemlerin halledilmesi olmal. Özellikle ML algoritmalar eksik verilere uyumlu deiller, bu yüzden ön ileme esnasnda kritik konulardan birisini bu ksm oluturuyor.

Konunun önem derecesi arttkça yaklamlarda deiiyor, genel bir yöntem ve herkesin kabul ettii bir yaklam yok fakat size en popüler olanlarn göstermeye çalacam.

Bu verileri direkt olarak kaldrabildiiniz durumlar yukarda iledik, imdi gelin kaldrmak istemediimiz durumlarda neler yapabiliriz bunlara bakalm.

- Mean, Median, Frequent, Constant
- Enterpolasyon
- KNN

3.2.1 1. En kolay teknik

Manuel

```
In [18]: df_1 = df.copy()
         df_1["Yas"]

Out[18]: 0           24
         1           22
         2           24
         3           23
         4    bilinmiyor
         5           23
         Name: Yas, dtype: object

In [21]: df_1

Out[21]:         sim    Soyad   Yas     Sehir
          0   Sevdanur     Genc  24.0     Bursa
          1     Selcuk     Genc  22.0    Ankara
          2    Huseyin    Sahin  24.0  Istanbul
          3      Dogus      Can  23.0       NaN
          4  Haticenur  Nalbant   NaN     Izmir
          5     Meltem    Onder  23.0  Istanbul
```

```
In [25]: df_1["Yas"].replace("bilinmiyor", np.nan, inplace=True) # ilk önce eksik veriyi NaN f

         df_1
```

Out[25]:

	sim	Soyad	Yas	Sehir
0	Sevdanur	Genc	24.0	Bursa
1	Selcuk	Genc	22.0	Ankara
2	Huseyin	Sahin	24.0	Istanbul
3	Dogus	Can	23.0	NaN
4	Haticenur	Nalbant	23.2	Izmir
5	Meltem	Onder	23.0	Istanbul

```
In [26]: df_1["Yas"].fillna(value=df_1["Yas"].mean(), inplace=True) # sonrasnda o kolonun orta

         df_1

         # na olan degerleri istenen sekilde dolduruyor.
```

Out[26]:

	sim	Soyad	Yas	Sehir
0	Sevdanur	Genc	24.0	Bursa
1	Selcuk	Genc	22.0	Ankara
2	Huseyin	Sahin	24.0	Istanbul
3	Dogus	Can	23.0	NaN
4	Haticenur	Nalbant	23.2	Izmir
5	Meltem	Onder	23.0	Istanbul

Scikit Scikit ile bu ilem oldukça kolaylatrlm, tekniinize göre 4 yöntem seçebiliyorsunuz.

- **mean:** Ortalama deer impute edilir.
- **median:** Medyan impute edilir.
- **most_frequent:** En çok tekrar eden deer eklenir.
- **constant:** sabit bir deer eklenir.

```
In [28]: import numpy as np
         from sklearn.impute import SimpleImputer

         # SimpleImputer : verilerdeki eksiklikleri verilen yontemle gidermek icin kullanilir.
         # missing_values : kayip verileri bul
         # Strategy : yukaridaki seceneklerden birisi yaziliyor, mean gibi.
         # fit_transform : bu islemi nereye yapacagini belirliyoruz.

         df_2 = df.copy()

         df_2["Yas"].replace("bilinmiyor", np.nan, inplace=True)

         imp_mean = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
         df_2["Yas"] = imp_mean.fit_transform(df_2[["Yas"]])

         df_2
```

```
Out[28]:          sim    Soyad   Yas     Sehir
          0   Sevdanur     Genc  24.0     Bursa
          1     Selcuk     Genc  22.0    Ankara
          2    Huseyin    Sahin  24.0  Istanbul
          3      Dogus      Can  23.0       NaN
          4  Haticenur  Nalbant  23.0     Izmir
          5     Meltem    Onder  23.0  Istanbul

In [29]: df = df_2.copy() # bir sonraki ileme geçmeden DataFrame'in son halini geri alyorum.
```

3.2.2 2. Enterpolasyon

Bu teknik biraz trickli olabilir, çünkü sürekli olduunuz bir veride kullanmanz mantkl olacaktr. Interpolasyon, elinizdeki veri noktalarnn arasnda bir deeri bilmediiniz, bu iki deer arasndaki bilinmeyin noktadaki deeri bulmanz salar. Mesela elinizde scaklk ile alakal time-series bir data olduunu düünelim burada bir eksik veriniz varsa bu iki nokta arasndaki deeri bulmak için kullanabilirsiniz. Aç/Tork grafii için verinin frekansn artrmak veya çözünürlük yükseltmek için kullanabilirsiniz.

Interpolasyon için basitçe bir örnee göz atalm:

- Sral giden bir array'de 2 deerinin eksik olduunu görüyorsunuz, lineer bir düzlemde 1 ve 3 says arasnda 2 olmas gerekmektedir.

Not: Interpolasyon'u yüksek dereceli polinomlar üzerinde de kullanabilirsiniz.

```
In [31]: s = pd.Series([0, 1, np.nan, 3])

         print(s)

0    0.0
1    1.0
2    NaN
3    3.0
dtype: float64


In [33]: s.interpolate()

Out[33]: 0    0.0
         1    1.0
         2    2.0
         3    3.0
         dtype: float64
```

3.2.3 3. En yakn komular

Varsaylan olarak, `nan_euclidean_distances` yakn komular bulmak için eksik deerleri destekleyen bir öklid mesafesi metrii kullanlr.

Her eksik özellii, `n_neighbors` says kadar olan yakn komularn deerleri kullanlarak bulunur.

Komularn özelliklerinin her bir komuya olan uzaklnn arlkl ortalamas alnr.

```
In [34]: import numpy as np
         from sklearn.impute import KNNImputer

         X = [[1, 2, np.nan], [3, 4, 3], [np.nan, 6, 5], [8, 8, 7]]
         pd.DataFrame(X)
```

Out[34]:

	0	1	2
0	1.0	2	NaN
1	3.0	4	3.0
2	NaN	6	5.0
3	8.0	8	7.0

```
In [35]: imputer = KNNImputer(n_neighbors=2, weights="uniform")
         X = imputer.fit_transform(X)
         pd.DataFrame(X)
```

Out[35]:

	0	1	2
0	1.0	2.0	4.0
1	3.0	4.0	3.0
2	5.5	6.0	5.0
3	8.0	8.0	7.0

4 3. Adm: Eksikleri tamamlayn!

Gördüünüz gibi matematiksel ve teorik ileri hallettikten sonra, **domain expert'in** kendi bilgisiyle ve kararlaryla tamamlamas gereken konular kalacaktr.

Örnek olarak aada `Sehir` kolonunda kalan bir eksiimiz var. Burada bir karar yukardaki tekniklerden birini kullanmaktr. Baka bir yaklam olarak burada bilinmeyen ehirlere `dier` yazabiliriz.

```
In [37]: df
```

Out[37]:

	sim	Soyad	Yas	Sehir
0	Sevdanur	Genc	24.0	Bursa
1	Selcuk	Genc	22.0	Ankara
2	Huseyin	Sahin	24.0	Istanbul
3	Dogus	Can	23.0	NaN
4	Haticenur	Nalbant	23.0	Izmir
5	Meltem	Onder	23.0	Istanbul

```
In [38]: df["Sehir"] = df["Sehir"].replace(np.nan, "dier")
         df
```

Out[38]:

	sim	Soyad	Yas	Sehir
0	Sevdanur	Genc	24.0	Bursa
1	Selcuk	Genc	22.0	Ankara
2	Huseyin	Sahin	24.0	Istanbul
3	Dogus	Can	23.0	dier
4	Haticenur	Nalbant	23.0	Izmir
5	Meltem	Onder	23.0	Istanbul

4.1 1. Standardization

Machine learning algoritmalarnn büyük bir çounluu iyi bir örenme için verinin standartlatrlmas gereklilii duyar. Eer veriniz Standart bir dalm göstermiyorsa, bu modelin örenmesinde kötü bir performansa sebep olabilecek etkiler dourabilir. Bu yüzden modele veriyi vermeden önce bir takm ön ilemler ile bu kötü etki ortadan kaldrlmas gerekmektedir.

4.1.1 1.1 Standard Scaler

```
In [39]: from sklearn.preprocessing import StandardScaler
         import numpy as np

         # bir column'daki dagilimin ortalamasi sifir, standart sapmasi bir
         # olacak sekilde yeniden scale etme islemine deniyor.

         df_ss = df.copy()

         df_ss["Yas_Scaled"] = StandardScaler().fit_transform(df_ss[["Yas"]])

         df_ss
```

Out[39]:

	sim	Soyad	Yas	Sehir	Yas_Scaled
0	Sevdanur	Genc	24.0	Bursa	1.212678
1	Selcuk	Genc	22.0	Ankara	-1.697749
2	Huseyin	Sahin	24.0	Istanbul	1.212678
3	Dogus	Can	23.0	dier	-0.242536
4	Haticenur	Nalbant	23.0	Izmir	-0.242536
5	Meltem	Onder	23.0	Istanbul	-0.242536

```
In [40]: print("X_train:", df_ss["Yas"].mean(axis=0), df_ss["Yas"].std(axis=0))

         print("X_scaled:", df_ss["Yas_Scaled"].mean(axis=0), df_ss["Yas_Scaled"].std(axis=0))
```

```
X_train: 23.166666666666668 0.752772652709081
X_scaled: -1.6930901125533637e-15 1.0954451150103321
```

4.1.2 1.2 MinMax Scaler

Eer çok küçük *standard sapmas* olan, küçük say deerleriyle çalyorsanz **MinMaxScaler** yararl olacaktr.

```
In [42]: from sklearn.preprocessing import MinMaxScaler
         import numpy as np

         df_mm = df.copy()

         df_mm["Yas_Scaled"] = MinMaxScaler().fit_transform(df_mm[["Yas"]])

         df_mm
```

```
Out[42]:         sim    Soyad   Yas     Sehir  Yas_Scaled
          0   Sevdanur     Genc  24.0     Bursa         1.0
          1     Selcuk     Genc  22.0    Ankara         0.0
          2    Huseyin    Sahin  24.0  Istanbul         1.0
          3      Dogus      Can  23.0     dier         0.5
          4  Haticenur  Nalbant  23.0     Izmir         0.5
          5     Meltem    Onder  23.0  Istanbul         0.5
```

4.2 Not:

Verilerinizde aykr deerler varken, scaling ilemleri çok iyi sonuçlar vermez.

Peki neden? Elinizdeki verinin 1 ile 10 arasnda dalm olduunu düünelim, veri setinin içerisinde yanl olarak yazlm 1000 deeri sizin scaling ileminizi bozarak, verinizi 1, 10 arasndaki tüm deerleri çok küçük bir alana sktracaktr.

```
In [43]: # KONUDAN BAIMSIZ
         # bir sonraki adm için dataframe'i son haline getiriyorum
         df["Yas"] = StandardScaler().fit_transform(df[["Yas"]])
```

4.3 3. Kategorik Deerlerin Ayrtrlmas

4.3.1 3.1 Label Encoding

Bir kolonunuzdaki deerleri sral bir biçimde saysal forma getirmek için kullanlr. Elinizde 4 adet ehir ismi olduunu varsayalm, eer bu deerler birçok satrda ayn isimlerle tekrarlanyorsa, bunlar saylar ile temsil edebilirsiniz. Aadaki örnekte görebileceiniz gibi Bursa 1 says ile, Ankara 0 ile, Istanbul 2 iletemsil edilecektir.

inverse_transform fonksiyonu ile geri alnabilir.

```
In [47]: df
```

```
Out[47]:         sim    Soyad       Yas     Sehir
          0   Sevdanur     Genc  1.212678     Bursa
          1     Selcuk     Genc -1.697749    Ankara
          2    Huseyin    Sahin  1.212678  Istanbul
          3      Dogus      Can -0.242536     dier
          4  Haticenur  Nalbant -0.242536     Izmir
          5     Meltem    Onder -0.242536  Istanbul
```

```
In [48]: from sklearn import preprocessing
         # from sklearn.preprocessing import LabelEncoder
         # le = LabelEncoder()

         df_le = df.copy()

         le = preprocessing.LabelEncoder()
         le.fit(df_le["Sehir"])

         list(le.classes_)
```

```
Out[48]: ['Ankara', 'Bursa', 'Istanbul', 'Izmir', 'dier']
```

```
In [49]: df_le["Sehir"] = le.transform(df_le["Sehir"])
         df_le
```

```
Out[49]:         sim    Soyad       Yas  Sehir
         0   Sevdanur     Genc  1.212678      1
         1     Selcuk     Genc -1.697749      0
         2    Huseyin    Sahin  1.212678      2
         3      Dogus      Can -0.242536      4
         4  Haticenur  Nalbant -0.242536      3
         5     Meltem    Onder -0.242536      2
```

```
In [50]: # Inverse_transform fonksiyonu ile geri alnabilir
         list(le.inverse_transform([2, 2, 1, 0]))
```

```
Out[50]: ['Istanbul', 'Istanbul', 'Bursa', 'Ankara']
```

4.3.2 3.2 One Hot Encoding

One Hot Encoding yöntemi bir kolon üzerindeki her bir snf, o snfn **unique** deerleri uzunluunda bir **vektöre** dönütürür. Her deer bu vektör üzerindeki yerini 1 saysn alarak belli eder, tanm daha iyi anlamak için örnee bakalm.

Eer kolonda [a, b, c] deerleri varsa. a [1, 0, 0] olarak temsil edilir, keza ayn ekilde b [0, 1, 0] eklinde temsil edilecekt.

One Hot encoding yöntemini **Sci-kit** yerine pandasn **get_dummies** fonksiyonu ile çok daha hzl ve rahat bir ekilde kullanabilirsiniz.

```
In [53]: import pandas as pd

         pd.get_dummies(df["Sehir"])
```

```
Out[53]:    Ankara  Bursa  Istanbul  Izmir  dier
         0       0      1         0      0      0
         1       1      0         0      0      0
         2       0      0         1      0      0
         3       0      0         0      0      1
         4       0      0         0      1      0
         5       0      0         1      0      0
```

```
In [ ]: #birden fazla kolon uzerinde calisip o kolonlari tek bir tabloda birlestirmek icin bu
        #pd.get_dummies(df_2, columns=['kolon1', 'kolon2', 'kolon3', 'kolon4'])
```

4.3.3 4. Kuantizasyon veya Binning

Kuantizasyon aslna bakarsanz, haberleme, sinyal ve elektronik derslerindeki önemli unsurlardan bir tanesidir. Bildiiniz gibi veri genellikle iki formda bulunur. Bunlardan ilki **ayrk** *(Discrete)* ve ikincisi **sürekli** (Continuous). Bazen verinizi snflara ayrmak istediinizde bu ilem çok büyük önem arz etmektedir. Sürekli bir deeri snflara ayrmak karar aaçlarnda veya hedefinizi snflandrmak istediinizde kullanabileceiniz bir fonksiyondur.

Burada en basit yöntem yuvarlama olabilir, sayy belirli sayların katlarına basitçe yuvarlayabilirsiniz, fakat daha bilimsel bir yöntem olan K-Bins kullanlabilir.

```
In [56]: X = np.array([[ -3., 5., 15 ],
                       [  0., 6., 14 ],
                       [  6., 3., 11 ]])
         # from sklearn import preprocessing
         preprocessing.KBinsDiscretizer(n_bins=[3, 2, 2], encode='ordinal').fit_transform(X)
         # nbins columnlari istenen sayida sinifa ayir.
         # ornegin 15,14,11 i 13 uzerindeki degerler bir sinif ve 1 ile gosterilsin,
         # 13 altindakiler bir sinif ve 0 ile gosterilsin sekilde nbins'te verilen degere
         # yani iki sinifa bolunmus oldu.

Out[56]: array([[0., 1., 1.],
                [1., 1., 1.],
                [2., 0., 0.]])
```

Örnei daha iyi anlamak adna her bir kolona bakabilirsiniz. **n_bins** parametresiyle kaç adet snfa bölmek istediinizi seçebilirsiniz. Fonksiyon her bir kolona bakarak, n_bins says kadar snfa bölecek ve deerlerin hangi snfa ait olduunu bularak bu sayyla temsil edecektir.

```
In [59]: binarizer = preprocessing.Binarizer(threshold=1.1)
         binarizer.transform(X)

         # verilen degerin uzerindeki degerlere 1, altindaki degerlere 0 degerini veriyor.

Out[59]: array([[0., 1., 1.],
                [0., 1., 1.],
                [1., 1., 1.]])
```

5 Feature Selection

Modelinizin iyi bir performans göstermesi için boyutsallnn azaltlmas ve güçlü ilikilere sahip parametrelerin, performans kötü etkileyecek dier parametrelerden ayrlmas gerekir. Çünkü bu öznitelikler (features) modele bir bilgi getirmiyor olabilirler.

Pekala boyut düürmenin veya öznitelik azaltmann yararlar nedir:

- Daha yüksek doruluk oran
- Overfitting probleminin önüne geçmek.
- Model eitim süresinin ksaltlmas.
- Daha etkin bir görselletirme
- Daha açklanabilir bir model.

5.1 Veri Seti

```
In [62]: import pandas as pd
         from sklearn.preprocessing import LabelEncoder
         from sklearn.preprocessing import StandardScaler
```

```
from sklearn.model_selection import train_test_split

data = pd.read_csv("mushrooms.csv")
data.head()
```

Out[62]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	\
0	p	x	s	n	t	p	f	
1	e	x	s	y	t	a	f	
2	e	b	s	w	t	l	f	
3	p	x	y	w	t	p	f	
4	e	x	s	g	f	n	f	

	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	\
0	c	n	k	...	s	
1	c	b	k	...	s	
2	c	b	n	...	s	
3	c	n	n	...	s	
4	w	b	k	...	s	

	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	\
0	w	w	p	w	
1	w	w	p	w	
2	w	w	p	w	
3	w	w	p	w	
4	w	w	p	w	

	ring-number	ring-type	spore-print-color	population	habitat
0	o	p	k	s	u
1	o	p	n	n	g
2	o	p	n	n	m
3	o	p	k	s	u
4	o	e	n	a	g

[5 rows x 23 columns]

In [64]:
```
X = data.drop(['class'], axis=1)
y = data['class']
```

In [66]:
```
X_encoded = pd.get_dummies(X, prefix_sep="_")
X_encoded
```

Out[66]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	1	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
5	0	0	0	0	0	
6	1	0	0	0	0	

7	1	0	0	0	0
8	0	0	0	0	0
9	1	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	1	0	0	0	0
13	0	0	0	0	0
14	0	0	0	0	0
15	0	0	0	0	1
16	0	0	1	0	0
17	0	0	0	0	0
18	0	0	0	0	0
19	0	0	0	0	0
20	1	0	0	0	0
21	0	0	0	0	0
22	1	0	0	0	0
23	1	0	0	0	0
24	1	0	0	0	0
25	0	0	1	0	0
26	0	0	0	0	0
27	0	0	0	0	0
28	0	0	1	0	0
29	0	0	0	0	0
...	...	...	...	...	...
8094	1	0	0	0	0
8095	0	0	0	0	0
8096	0	0	0	1	0
8097	0	0	0	1	0
8098	0	0	0	1	0
8099	0	0	0	1	0
8100	0	0	1	0	0
8101	0	0	0	1	0
8102	0	0	0	0	0
8103	0	0	0	1	0
8104	0	0	0	1	0
8105	0	0	0	1	0
8106	0	0	0	1	0
8107	0	0	0	0	0
8108	0	0	0	1	0
8109	1	0	0	0	0
8110	0	0	0	0	0
8111	0	0	0	1	0
8112	0	0	0	1	0
8113	0	0	0	1	0
8114	0	0	1	0	0
8115	0	0	0	0	0
8116	0	0	0	1	0
8117	0	0	0	1	0

8118	0	0	0	1	0
8119	0	0	0	1	0
8120	0	0	0	0	0
8121	0	0	1	0	0
8122	0	0	0	1	0
8123	0	0	0	0	0

	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	\
0	1	0	0	1	0	
1	1	0	0	1	0	
2	0	0	0	1	0	
3	1	0	0	0	1	
4	1	0	0	1	0	
5	1	0	0	0	1	
6	0	0	0	1	0	
7	0	0	0	0	1	
8	1	0	0	0	1	
9	0	0	0	1	0	
10	1	0	0	0	1	
11	1	0	0	0	1	
12	0	0	0	1	0	
13	1	0	0	0	1	
14	1	1	0	0	0	
15	0	1	0	0	0	
16	0	1	0	0	0	
17	1	0	0	1	0	
18	1	0	0	0	1	
19	1	0	0	1	0	
20	0	0	0	1	0	
21	1	0	0	0	1	
22	0	0	0	0	1	
23	0	0	0	0	1	
24	0	0	0	1	0	
25	0	0	0	1	0	
26	1	0	0	0	1	
27	1	0	0	0	1	
28	0	1	0	0	0	
29	1	0	0	1	0	
...	...	...	...	...	...	
8094	0	0	0	1	0	
8095	1	0	0	0	1	
8096	0	1	0	0	0	
8097	0	0	0	0	1	
8098	0	0	0	1	0	
8099	0	1	0	0	0	
8100	0	0	0	1	0	
8101	0	0	0	1	0	
8102	1	0	0	1	0	

8103	0	0	0	1	0
8104	0	0	0	1	0
8105	0	0	0	1	0
8106	0	0	0	1	0
8107	1	0	0	1	0
8108	0	0	0	0	1
8109	0	0	0	1	0
8110	1	0	0	1	0
8111	0	0	0	1	0
8112	0	0	0	1	0
8113	0	0	0	0	1
8114	0	0	0	0	1
8115	1	0	0	1	0
8116	0	0	0	0	1
8117	0	0	0	1	0
8118	0	0	0	0	1
8119	0	0	0	1	0
8120	1	0	0	1	0
8121	0	0	0	1	0
8122	0	0	0	0	1
8123	1	0	0	1	0

	...	population_s	population_v	population_y	habitat_d	habitat_g	\
0	...	1	0	0	0	0	
1	...	0	0	0	0	1	
2	...	0	0	0	0	0	
3	...	1	0	0	0	0	
4	...	0	0	0	0	1	
5	...	0	0	0	0	1	
6	...	0	0	0	0	0	
7	...	1	0	0	0	0	
8	...	0	1	0	0	1	
9	...	1	0	0	0	0	
10	...	0	0	0	0	1	
11	...	1	0	0	0	0	
12	...	1	0	0	0	1	
13	...	0	1	0	0	0	
14	...	0	0	0	0	1	
15	...	0	0	1	0	0	
16	...	0	0	0	0	1	
17	...	1	0	0	0	1	
18	...	1	0	0	0	0	
19	...	1	0	0	0	0	
20	...	1	0	0	0	0	
21	...	0	1	0	0	1	
22	...	1	0	0	0	0	
23	...	0	0	0	0	0	
24	...	1	0	0	0	0	

25	...	0	1	0	0	1
26	...	0	0	0	0	0
27	...	0	0	0	0	0
28	...	0	0	1	0	0
29	...	0	1	0	1	0
...	...	...	...	...	...	...
8094	...	0	0	0	0	1
8095	...	0	0	0	1	0
8096	...	0	0	0	0	1
8097	...	0	1	0	0	0
8098	...	0	1	0	1	0
8099	...	1	0	0	0	1
8100	...	0	1	0	0	0
8101	...	0	1	0	0	0
8102	...	0	0	0	0	0
8103	...	0	0	0	0	0
8104	...	0	1	0	0	0
8105	...	0	1	0	0	0
8106	...	0	1	0	0	0
8107	...	0	0	0	0	0
8108	...	0	1	0	0	0
8109	...	0	0	0	0	1
8110	...	0	1	0	0	0
8111	...	0	0	0	0	1
8112	...	0	1	0	0	0
8113	...	0	1	0	1	0
8114	...	0	0	0	1	0
8115	...	0	1	0	0	0
8116	...	0	1	0	0	0
8117	...	0	1	0	1	0
8118	...	0	1	0	1	0
8119	...	0	0	0	0	0
8120	...	0	1	0	0	0
8121	...	0	0	0	0	0
8122	...	0	1	0	0	0
8123	...	0	0	0	0	0

	habitat_l	habitat_m	habitat_p	habitat_u	habitat_w
0	0	0	0	1	0
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	1	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	1	0	0	0
7	0	1	0	0	0
8	0	0	0	0	0
9	0	1	0	0	0

10	0	0	0	0	0
11	0	1	0	0	0
12	0	0	0	0	0
13	0	0	0	1	0
14	0	0	0	0	0
15	0	0	0	1	0
16	0	0	0	0	0
17	0	0	0	0	0
18	0	0	0	1	0
19	0	0	0	1	0
20	0	1	0	0	0
21	0	0	0	0	0
22	0	1	0	0	0
23	0	1	0	0	0
24	0	1	0	0	0
25	0	0	0	0	0
26	0	1	0	0	0
27	0	1	0	0	0
28	0	0	0	1	0
29	0	0	0	0	0
...	...	...	...	...	...
8094	0	0	0	0	0
8095	0	0	0	0	0
8096	0	0	0	0	0
8097	1	0	0	0	0
8098	0	0	0	0	0
8099	0	0	0	0	0
8100	1	0	0	0	0
8101	0	0	1	0	0
8102	1	0	0	0	0
8103	1	0	0	0	0
8104	1	0	0	0	0
8105	1	0	0	0	0
8106	1	0	0	0	0
8107	1	0	0	0	0
8108	1	0	0	0	0
8109	0	0	0	0	0
8110	1	0	0	0	0
8111	0	0	0	0	0
8112	1	0	0	0	0
8113	0	0	0	0	0
8114	0	0	0	0	0
8115	1	0	0	0	0
8116	1	0	0	0	0
8117	0	0	0	0	0
8118	0	0	0	0	0
8119	1	0	0	0	0
8120	1	0	0	0	0

```
8121          1          0          0          0          0
8122          1          0          0          0          0
8123          1          0          0          0          0

[8124 rows x 117 columns]
```

```
In [68]: y_encoded = LabelEncoder().fit_transform(y)
         y_encoded
```

```
Out[68]: array([1, 0, 0, ..., 0, 1, 0])
```

```
In [72]: X_scaled = StandardScaler().fit_transform(X_encoded)
         X_scaled
```

```
Out[72]: array([[-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                  4.59086996, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [ 4.11988487, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                ...,
                [-0.24272523, -0.02219484,  1.2559503 , ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197]])
```

```
In [78]: X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size = (

         # veriyi ikiye boluyoruz. yuzde 30 a boluyor. degerler random olarak siralaniyor.
```

```
In [79]: X_train
```

```
Out[79]: array([[-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                ...,
                [-0.24272523, -0.02219484,  1.2559503 , ..., -0.40484176,
                  4.59086996, -0.15558197],
                [-0.24272523, -0.02219484,  1.2559503 , ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484,  1.2559503 , ...,  2.47010093,
                 -0.21782364, -0.15558197]])
```

```
In [80]: X_test
```

```
Out[80]: array([[-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ...,  2.47010093,
                 -0.21782364, -0.15558197],
                ...,
                [-0.24272523, -0.02219484,  1.2559503 , ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197],
                [-0.24272523, -0.02219484, -0.79620985, ..., -0.40484176,
                 -0.21782364, -0.15558197]])

In [81]: y_train

Out[81]: array([0, 0, 0, ..., 1, 0, 1])

In [82]: y_test

Out[82]: array([1, 0, 1, ..., 0, 0, 1])
```

5.2 Feature Importance

Karar aaçlar çeitli özniteliklerin önem derecelerini sralamak için kullanlabilir. Karar aaçlarndaki dallanma bildiiniz gibi özniteliklerin snflandrclyla belirlenir. Bu yüzden daha çok kullanlan nodelar daha yüksek öneme sahip olabilirler.

```
In [84]: import time
         from sklearn.metrics import classification_report, confusion_matrix
         from sklearn.ensemble import RandomForestClassifier

In [89]: start = time.process_time()

         model = RandomForestClassifier(n_estimators=700).fit(X_train, y_train)

         print(time.process_time() - start)

4.28125


In [90]: preds = model.predict(X_test)

         print(confusion_matrix(y_test, preds))

[[1274    0]
 [   0 1164]]


In [91]: print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1274
           1       1.00      1.00      1.00      1164

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```

Tam bir baar oranna sahibiz fakat burada bakacamz konu aslnda hangi niteliklerin ne kadar önemli olduu. Bu yüzden feature importance metoduyla eitilmi modelin en önemli olduu 10 parametreyi görselletiriyorum.

```
In [95]: import matplotlib.pyplot as plt
         from matplotlib.pyplot import figure

         model.feature_importances_
```

```
Out[95]: array([2.64563348e-03, 7.59918070e-05, 7.73758089e-04, 4.40006468e-04,
                6.08874560e-04, 1.00832649e-03, 4.32832441e-03, 1.64976352e-04,
                4.24285400e-03, 1.61588342e-03, 2.58886586e-03, 5.73825870e-04,
                7.82394865e-04, 1.14767285e-03, 1.39353149e-03, 1.58810121e-03,
                2.22643890e-04, 1.77454105e-04, 2.65351229e-03, 3.69905014e-03,
                3.07031375e-02, 3.22883144e-02, 5.56508199e-03, 1.11486859e-02,
                7.13903147e-02, 5.25457929e-03, 7.69483546e-04, 1.23607050e-01,
                1.44119822e-02, 4.09440818e-03, 4.42875548e-03, 1.23084092e-03,
                1.32188221e-03, 2.24407769e-02, 1.96390741e-02, 5.72483638e-02,
                5.80515180e-02, 3.88036019e-02, 2.22614791e-04, 8.52542227e-04,
                1.68174350e-03, 2.97597585e-04, 1.63602587e-03, 1.25659560e-04,
                3.14334681e-04, 1.94027878e-03, 5.02829236e-04, 2.35154721e-03,
                8.45607304e-05, 9.92902299e-03, 1.03725861e-02, 9.06692654e-03,
                1.52624536e-02, 9.10169336e-03, 1.78015320e-02, 1.63924270e-03,
                4.83931464e-03, 4.09703127e-02, 2.13712582e-02, 4.24795950e-04,
                5.23883953e-03, 3.58306720e-02, 1.45075926e-02, 2.93102041e-03,
                1.15606343e-03, 6.88584593e-04, 1.00109692e-04, 8.22432203e-04,
                1.05686049e-03, 1.47601883e-03, 1.09396414e-03, 5.95534841e-03,
                3.27415857e-04, 1.73849162e-03, 7.39998964e-04, 1.26418058e-04,
                1.26157104e-03, 2.06967954e-03, 1.07812798e-03, 8.65851798e-04,
                4.42685176e-03, 9.27334661e-04, 0.00000000e+00, 2.35249485e-04,
                2.58757796e-04, 1.10015247e-03, 2.71770667e-04, 6.58725421e-04,
                5.10524617e-03, 7.32443224e-03, 7.97399620e-03, 1.40880424e-03,
                2.05172053e-02, 7.27260527e-04, 3.03209680e-02, 7.83428999e-07,
                4.19918494e-02, 8.22398795e-03, 1.00397589e-02, 6.16087638e-05,
                7.42623191e-03, 1.38677325e-03, 1.32000676e-02, 5.62249921e-05,
                1.54493588e-03, 1.44337123e-03, 2.50693713e-03, 3.39621300e-03,
                2.75408501e-02, 4.31604125e-03, 8.20887544e-03, 7.39812547e-03,
```

```
       1.15282404e-03, 2.85894215e-03, 4.01490817e-03, 7.81254477e-03,
       1.17792316e-03])
```

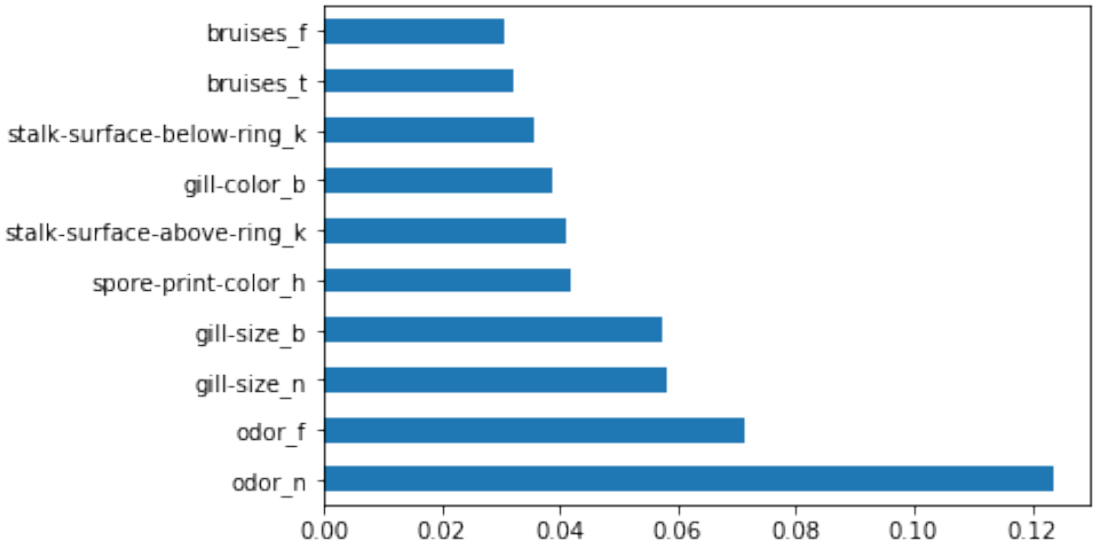
In [100]:
```
feature_imp = pd.Series(model.feature_importances_, index= X_encoded.columns)
feature_imp
```

Out[100]:
```
cap-shape_b            2.645633e-03
cap-shape_c            7.599181e-05
cap-shape_f            7.737581e-04
cap-shape_k            4.400065e-04
cap-shape_s            6.088746e-04
cap-shape_x            1.008326e-03
cap-surface_f          4.328324e-03
cap-surface_g          1.649764e-04
cap-surface_s          4.242854e-03
cap-surface_y          1.615883e-03
cap-color_b            2.588866e-03
cap-color_c            5.738259e-04
cap-color_e            7.823949e-04
cap-color_g            1.147673e-03
cap-color_n            1.393531e-03
cap-color_p            1.588101e-03
cap-color_r            2.226439e-04
cap-color_u            1.774541e-04
cap-color_w            2.653512e-03
cap-color_y            3.699050e-03
bruises_f              3.070314e-02
bruises_t              3.228831e-02
odor_a                 5.565082e-03
odor_c                 1.114869e-02
odor_f                 7.139031e-02
odor_l                 5.254579e-03
odor_m                 7.694835e-04
odor_n                 1.236071e-01
odor_p                 1.441198e-02
odor_s                 4.094408e-03
                           ...
ring-number_n          6.587254e-04
ring-number_o          5.105246e-03
ring-number_t          7.324432e-03
ring-type_e            7.973996e-03
ring-type_f            1.408804e-03
ring-type_l            2.051721e-02
ring-type_n            7.272605e-04
ring-type_p            3.032097e-02
spore-print-color_b    7.834290e-07
spore-print-color_h    4.199185e-02
spore-print-color_k    8.223988e-03
```

```
spore-print-color_n    1.003976e-02
spore-print-color_o    6.160876e-05
spore-print-color_r    7.426232e-03
spore-print-color_u    1.386773e-03
spore-print-color_w    1.320007e-02
spore-print-color_y    5.622499e-05
population_a           1.544936e-03
population_c           1.443371e-03
population_n           2.506937e-03
population_s           3.396213e-03
population_v           2.754085e-02
population_y           4.316041e-03
habitat_d              8.208875e-03
habitat_g              7.398125e-03
habitat_l              1.152824e-03
habitat_m              2.858942e-03
habitat_p              4.014908e-03
habitat_u              7.812545e-03
habitat_w              1.177923e-03
Length: 117, dtype: float64
```

In [102]:
```
feature_imp.nlargest(10).plot(kind='barh')
# nlargest : girilen degerin en buyuk on degerini getiriyor
```

Out[102]: <matplotlib.axes._subplots.AxesSubplot at 0x230f4249b00>



In [104]:
```
best_feat = feature_imp.nlargest(4).index.to_list()
best_feat
```

```
Out[104]: ['odor_n', 'odor_f', 'gill-size_n', 'gill-size_b']

In [105]: X_reduced = X_encoded[feature_imp.nlargest(4).index]
          X_reduced
```

Out[105]:

	odor_n	odor_f	gill-size_n	gill-size_b
0	0	0	1	0
1	0	0	0	1
2	0	0	0	1
3	0	0	1	0
4	1	0	0	1
5	0	0	0	1
6	0	0	0	1
7	0	0	0	1
8	0	0	1	0
9	0	0	0	1
10	0	0	0	1
11	0	0	0	1
12	0	0	0	1
13	0	0	1	0
14	1	0	0	1
15	1	0	1	0
16	1	0	0	1
17	0	0	1	0
18	0	0	1	0
19	0	0	1	0
20	0	0	0	1
21	0	0	1	0
22	0	0	0	1
23	0	0	0	1
24	0	0	0	1
25	0	0	1	0
26	0	0	0	1
27	0	0	0	1
28	1	0	1	0
29	0	0	1	0
...	...	...	...	...
8094	1	0	0	1
8095	0	0	0	1
8096	1	0	0	1
8097	0	0	1	0
8098	0	0	1	0
8099	1	0	0	1
8100	1	0	0	1
8101	0	0	1	0
8102	1	0	0	1
8103	1	0	0	1
8104	1	0	0	1

```
8105       1       0            0            1
8106       1       0            0            1
8107       1       0            0            1
8108       0       0            1            0
8109       1       0            0            1
8110       1       0            0            1
8111       1       0            0            1
8112       1       0            0            1
8113       0       0            1            0
8114       0       0            0            1
8115       1       0            0            1
8116       0       0            1            0
8117       0       0            1            0
8118       0       1            1            0
8119       1       0            0            1
8120       1       0            0            1
8121       1       0            0            1
8122       0       0            1            0
8123       1       0            0            1

[8124 rows x 4 columns]
```

```
In [106]: Xr_scaled = StandardScaler().fit_transform(X_reduced)
          Xr_scaled
```

```
Out[106]: array([[-0.87614155, -0.60180814,  1.49468272, -1.49468272],
                 [-0.87614155, -0.60180814, -0.66903831,  0.66903831],
                 [-0.87614155, -0.60180814, -0.66903831,  0.66903831],
                 ...,
                 [ 1.14136808, -0.60180814, -0.66903831,  0.66903831],
                 [-0.87614155, -0.60180814,  1.49468272, -1.49468272],
                 [ 1.14136808, -0.60180814, -0.66903831,  0.66903831]])
```

```
In [107]: Xr_train, Xr_test, yr_train, yr_test = train_test_split(Xr_scaled, y, test_size = 0.3
                                                                  random_state = 101)
```

```
In [109]: start = time.process_time()
          rmodel = RandomForestClassifier(n_estimators=700).fit(Xr_train,yr_train)
          print(time.process_time() - start)
```

```
0.96875
```

```
In [110]: rpred = rmodel.predict(Xr_test)
          print(confusion_matrix(yr_test, rpred))
```

```
[[1248   26]
 [  53 1111]]
```

```
In [113]: print(classification_report(yr_test, rpred))
```

```
              precision    recall  f1-score   support

           e       0.96      0.98      0.97      1274
           p       0.98      0.95      0.97      1164

    accuracy                           0.97      2438
   macro avg       0.97      0.97      0.97      2438
weighted avg       0.97      0.97      0.97      2438
```

Çok açk bir ekilde görebiliriz ki, eitim süresi yar yarya inerken accuracy'den çok az kaybettik. Aslna bakarsanz bu çok küçük bir veriseti kazancmz 1 saniye kadar fakat bunu milyonlarca satra sahip bir verisetiyle saatlerce eittiiniz bir model olduunu düünürseniz kesinlikle gireceiniz bir tradeoff olacaktr.

5.3 Correlation Matrix

```
In [114]: import seaborn as sns

          X = data.drop(['class'], axis=1)
          y = data['class']

In [115]: X_encoded = pd.get_dummies(X, prefix_sep="_")
          X_encoded
```

Out[115]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	1	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
5	0	0	0	0	0	
6	1	0	0	0	0	
7	1	0	0	0	0	
8	0	0	0	0	0	
9	1	0	0	0	0	
10	0	0	0	0	0	
11	0	0	0	0	0	
12	1	0	0	0	0	
13	0	0	0	0	0	
14	0	0	0	0	0	
15	0	0	0	0	1	
16	0	0	1	0	0	
17	0	0	0	0	0	
18	0	0	0	0	0	
19	0	0	0	0	0	

20	1	0	0	0	0
21	0	0	0	0	0
22	1	0	0	0	0
23	1	0	0	0	0
24	1	0	0	0	0
25	0	0	1	0	0
26	0	0	0	0	0
27	0	0	0	0	0
28	0	0	1	0	0
29	0	0	0	0	0
...	...	...	...	...	...
8094	1	0	0	0	0
8095	0	0	0	0	0
8096	0	0	0	1	0
8097	0	0	0	1	0
8098	0	0	0	1	0
8099	0	0	0	1	0
8100	0	0	1	0	0
8101	0	0	0	1	0
8102	0	0	0	0	0
8103	0	0	0	1	0
8104	0	0	0	1	0
8105	0	0	0	1	0
8106	0	0	0	1	0
8107	0	0	0	0	0
8108	0	0	0	1	0
8109	1	0	0	0	0
8110	0	0	0	0	0
8111	0	0	0	1	0
8112	0	0	0	1	0
8113	0	0	0	1	0
8114	0	0	1	0	0
8115	0	0	0	0	0
8116	0	0	0	1	0
8117	0	0	0	1	0
8118	0	0	0	1	0
8119	0	0	0	1	0
8120	0	0	0	0	0
8121	0	0	1	0	0
8122	0	0	0	1	0
8123	0	0	0	0	0

	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	\
0	1	0	0	1	0	
1	1	0	0	1	0	
2	0	0	0	1	0	
3	1	0	0	0	1	
4	1	0	0	1	0	

5	1	0	0	0	1
6	0	0	0	1	0
7	0	0	0	0	1
8	1	0	0	0	1
9	0	0	0	1	0
10	1	0	0	0	1
11	1	0	0	0	1
12	0	0	0	1	0
13	1	0	0	0	1
14	1	1	0	0	0
15	0	1	0	0	0
16	0	1	0	0	0
17	1	0	0	1	0
18	1	0	0	0	1
19	1	0	0	1	0
20	0	0	0	1	0
21	1	0	0	0	1
22	0	0	0	0	1
23	0	0	0	0	1
24	0	0	0	1	0
25	0	0	0	1	0
26	1	0	0	0	1
27	1	0	0	0	1
28	0	1	0	0	0
29	1	0	0	1	0
...	...	...	...	...	...
8094	0	0	0	1	0
8095	1	0	0	0	1
8096	0	1	0	0	0
8097	0	0	0	0	1
8098	0	0	0	1	0
8099	0	1	0	0	0
8100	0	0	0	1	0
8101	0	0	0	1	0
8102	1	0	0	1	0
8103	0	0	0	1	0
8104	0	0	0	1	0
8105	0	0	0	1	0
8106	0	0	0	1	0
8107	1	0	0	1	0
8108	0	0	0	0	1
8109	0	0	0	1	0
8110	1	0	0	1	0
8111	0	0	0	1	0
8112	0	0	0	1	0
8113	0	0	0	0	1
8114	0	0	0	0	1
8115	1	0	0	1	0

8116	0	0	0	0	1
8117	0	0	0	1	0
8118	0	0	0	0	1
8119	0	0	0	1	0
8120	1	0	0	1	0
8121	0	0	0	1	0
8122	0	0	0	0	1
8123	1	0	0	1	0

	...	population_s	population_v	population_y	habitat_d	habitat_g	\
0	...	1	0	0	0	0	
1	...	0	0	0	0	1	
2	...	0	0	0	0	0	
3	...	1	0	0	0	0	
4	...	0	0	0	0	1	
5	...	0	0	0	0	1	
6	...	0	0	0	0	0	
7	...	1	0	0	0	0	
8	...	0	1	0	0	1	
9	...	1	0	0	0	0	
10	...	0	0	0	0	1	
11	...	1	0	0	0	0	
12	...	1	0	0	0	1	
13	...	0	1	0	0	0	
14	...	0	0	0	0	1	
15	...	0	0	1	0	0	
16	...	0	0	0	0	1	
17	...	1	0	0	0	1	
18	...	1	0	0	0	0	
19	...	1	0	0	0	0	
20	...	1	0	0	0	0	
21	...	0	1	0	0	1	
22	...	1	0	0	0	0	
23	...	0	0	0	0	0	
24	...	1	0	0	0	0	
25	...	0	1	0	0	1	
26	...	0	0	0	0	0	
27	...	0	0	0	0	0	
28	...	0	0	1	0	0	
29	...	0	1	0	1	0	
...	...	...	...	...	...	...	
8094	...	0	0	0	0	1	
8095	...	0	0	0	1	0	
8096	...	0	0	0	0	1	
8097	...	0	1	0	0	0	
8098	...	0	1	0	1	0	
8099	...	1	0	0	0	1	
8100	...	0	1	0	0	0	

8101	...	0	1	0	0	0
8102	...	0	0	0	0	0
8103	...	0	0	0	0	0
8104	...	0	1	0	0	0
8105	...	0	1	0	0	0
8106	...	0	1	0	0	0
8107	...	0	0	0	0	0
8108	...	0	1	0	0	0
8109	...	0	0	0	0	1
8110	...	0	1	0	0	0
8111	...	0	0	0	0	1
8112	...	0	1	0	0	0
8113	...	0	1	0	1	0
8114	...	0	0	0	1	0
8115	...	0	1	0	0	0
8116	...	0	1	0	0	0
8117	...	0	1	0	1	0
8118	...	0	1	0	1	0
8119	...	0	0	0	0	0
8120	...	0	1	0	0	0
8121	...	0	0	0	0	0
8122	...	0	1	0	0	0
8123	...	0	0	0	0	0

	habitat_l	habitat_m	habitat_p	habitat_u	habitat_w
0	0	0	0	1	0
1	0	0	0	0	0
2	0	1	0	0	0
3	0	0	0	1	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	1	0	0	0
7	0	1	0	0	0
8	0	0	0	0	0
9	0	1	0	0	0
10	0	0	0	0	0
11	0	1	0	0	0
12	0	0	0	0	0
13	0	0	0	1	0
14	0	0	0	0	0
15	0	0	0	1	0
16	0	0	0	0	0
17	0	0	0	0	0
18	0	0	0	1	0
19	0	0	0	1	0
20	0	1	0	0	0
21	0	0	0	0	0
22	0	1	0	0	0

```
23            0          1          0          0          0
24            0          1          0          0          0
25            0          0          0          0          0
26            0          1          0          0          0
27            0          1          0          0          0
28            0          0          0          1          0
29            0          0          0          0          0
...         ...        ...        ...        ...        ...
8094          0          0          0          0          0
8095          0          0          0          0          0
8096          0          0          0          0          0
8097          1          0          0          0          0
8098          0          0          0          0          0
8099          0          0          0          0          0
8100          1          0          0          0          0
8101          0          0          1          0          0
8102          1          0          0          0          0
8103          1          0          0          0          0
8104          1          0          0          0          0
8105          1          0          0          0          0
8106          1          0          0          0          0
8107          1          0          0          0          0
8108          1          0          0          0          0
8109          0          0          0          0          0
8110          1          0          0          0          0
8111          0          0          0          0          0
8112          1          0          0          0          0
8113          0          0          0          0          0
8114          0          0          0          0          0
8115          1          0          0          0          0
8116          1          0          0          0          0
8117          0          0          0          0          0
8118          0          0          0          0          0
8119          1          0          0          0          0
8120          1          0          0          0          0
8121          1          0          0          0          0
8122          1          0          0          0          0
8123          1          0          0          0          0

[8124 rows x 117 columns]
```

```
In [116]: y_encoded = LabelEncoder().fit_transform(y)
          y_encoded
```

```
Out[116]: array([1, 0, 0, ..., 0, 1, 0])
```

```
In [117]: X_encoded["Class"] = y_encoded
```

```
In [120]: X_encoded.iloc[:, -7:].corr()
```

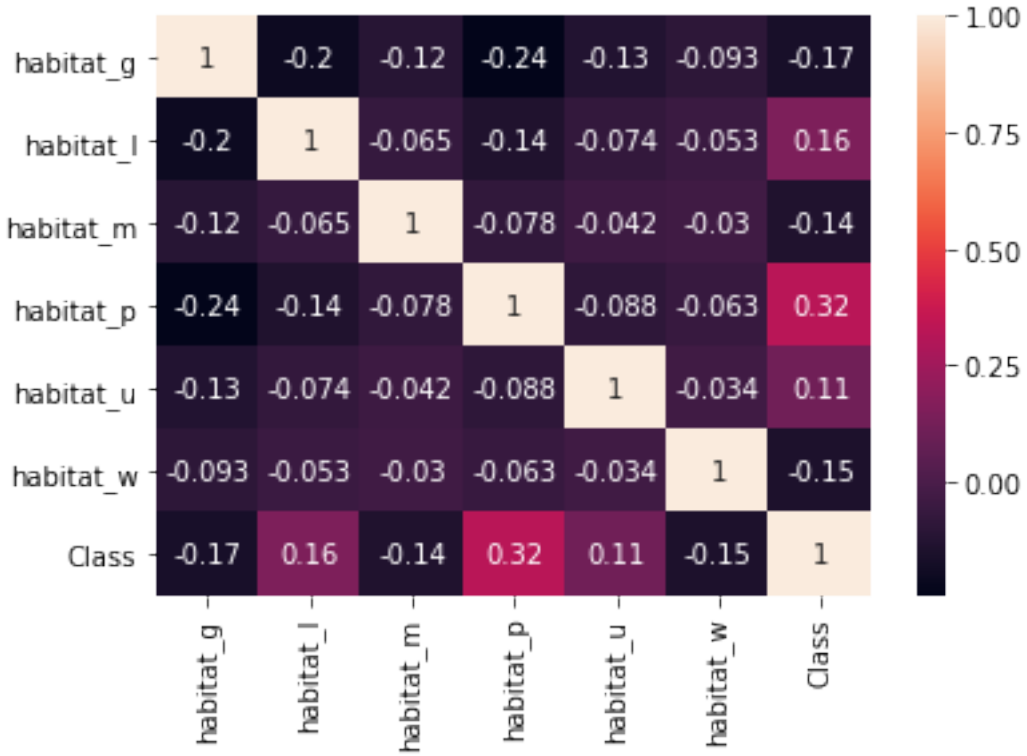
Out[120]:

	habitat_g	habitat_l	habitat_m	habitat_p	habitat_u	habitat_w	\
habitat_g	1.000000	-0.202512	-0.115762	-0.242715	-0.130592	-0.093276	
habitat_l	-0.202512	1.000000	-0.065222	-0.136749	-0.073577	-0.052553	
habitat_m	-0.115762	-0.065222	1.000000	-0.078170	-0.042059	-0.030041	
habitat_p	-0.242715	-0.136749	-0.078170	1.000000	-0.088184	-0.062986	
habitat_u	-0.130592	-0.073577	-0.042059	-0.088184	1.000000	-0.033889	
habitat_w	-0.093276	-0.052553	-0.030041	-0.062986	-0.033889	1.000000	
Class	-0.165004	0.155150	-0.138627	0.323346	0.112078	-0.150087	

	Class
habitat_g	-0.165004
habitat_l	0.155150
habitat_m	-0.138627
habitat_p	0.323346
habitat_u	0.112078
habitat_w	-0.150087
Class	1.000000

In [124]:
```
sns.heatmap(X_encoded.iloc[:, -7:].corr(), annot=True)
# annot : renklerin icerisinde sayilarin olup olmamasi ayarlaniyor.
```

Out[124]: <matplotlib.axes._subplots.AxesSubplot at 0x230f57909b0>



Belirttiimiz gibi eksi ve art deerler güçlü korelasyonu ifade ediyor, burada saynn pozitif ve negatif olmas ilikinin ters veya doru orantl olarak deimesi ile alakal, her ikisi de bizim için iyi featurelar olabilir bu yüzden dataframe'in mutlak deerini alarak en yüksek deerli olanlar getireceiz.

```
In [126]: X_encoded.corr().abs()["Class"]

          # .nlarget ile sral bir ekilde en yüksek 10 deeri alabiliriz.
          X_encoded.corr().abs()["Class"].nlargest(10)
```

```
Out[126]: Class                          1.000000
          odor_n                         0.785557
          odor_f                         0.623842
          stalk-surface-above-ring_k     0.587658
          stalk-surface-below-ring_k     0.573524
          ring-type_p                    0.540469
          gill-size_n                    0.540024
          gill-size_b                    0.540024
          gill-color_b                   0.538808
          bruises_f                      0.501530
          Name: Class, dtype: float64
```

Bu zamana kadar yazdmz ksmn sonunda index metodunu ekleyerek sadece kolon isimlerini istiyorum ve bunu ana datasetimizden baka bir deikene aktaryorum. Birazdan sadece bu ksm kullanyor olacaz, bu sayede daha okunakl ve en yüksek 10 korelasyon deerine sahip kolon ile birlikte çalyor olacaz.

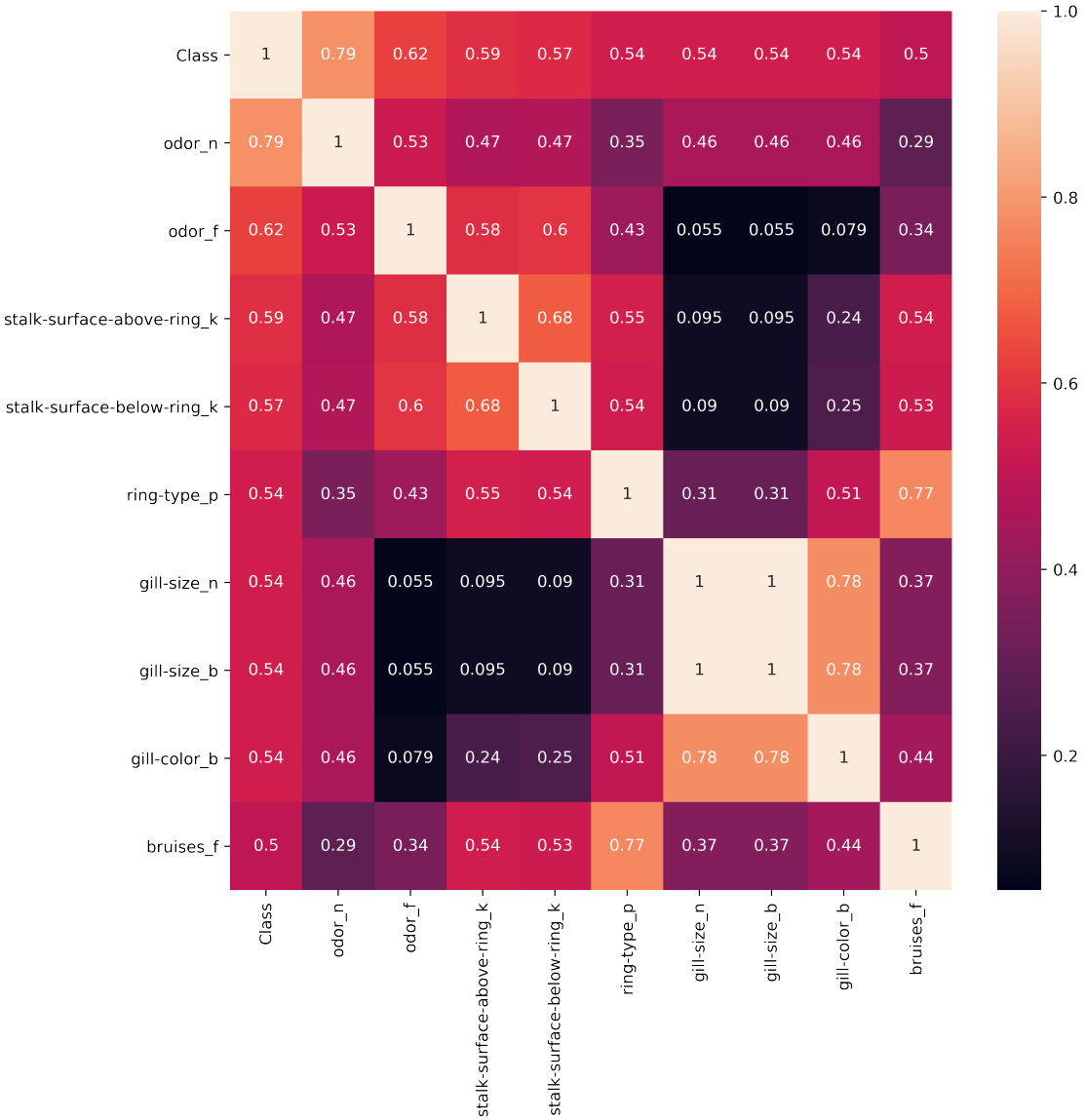
```
In [127]: X_reduced_col_names = X_encoded.corr().abs()["Class"].nlargest(10).index
          X_encoded[X_reduced_col_names].corr()
```

```
Out[127]:                                Class    odor_n    odor_f  \
          Class                       1.000000 -0.785557  0.623842
          odor_n                     -0.785557  1.000000 -0.527269
          odor_f                      0.623842 -0.527269  1.000000
          stalk-surface-above-ring_k  0.587658 -0.466499  0.584189
          stalk-surface-below-ring_k  0.573524 -0.471920  0.600449
          ring-type_p                -0.540469  0.352151 -0.427514
          gill-size_n                 0.540024 -0.457211 -0.055394
          gill-size_b                -0.540024  0.457211  0.055394
          gill-color_b                0.538808 -0.455399  0.079360
          bruises_f                   0.501530 -0.285171  0.344642

                                    stalk-surface-above-ring_k  \
          Class                                       0.587658
          odor_n                                     -0.466499
          odor_f                                      0.584189
          stalk-surface-above-ring_k                  1.000000
          stalk-surface-below-ring_k                  0.677074
          ring-type_p                                -0.549484
          gill-size_n                                 0.095225
```

```
gill-size_b                                -0.095225
gill-color_b                                0.237814
bruises_f                                   0.541494

                           stalk-surface-below-ring_k  ring-type_p  \
Class                                        0.573524    -0.540469
odor_n                                      -0.471920     0.352151
odor_f                                       0.600449    -0.427514
stalk-surface-above-ring_k                   0.677074    -0.549484
stalk-surface-below-ring_k                   1.000000    -0.536122
ring-type_p                                 -0.536122     1.000000
gill-size_n                                  0.089569    -0.308466
gill-size_b                                 -0.089569     0.308466
gill-color_b                                 0.249536    -0.507885
bruises_f                                    0.530549    -0.767036

                           gill-size_n  gill-size_b  gill-color_b  bruises_f
Class                         0.540024    -0.540024      0.538808   0.501530
odor_n                       -0.457211     0.457211     -0.455399  -0.285171
odor_f                       -0.055394     0.055394      0.079360   0.344642
stalk-surface-above-ring_k    0.095225    -0.095225      0.237814   0.541494
stalk-surface-below-ring_k    0.089569    -0.089569      0.249536   0.530549
ring-type_p                  -0.308466     0.308466     -0.507885  -0.767036
gill-size_n                   1.000000    -1.000000      0.776903   0.369596
gill-size_b                  -1.000000     1.000000     -0.776903  -0.369596
gill-color_b                  0.776903    -0.776903      1.000000   0.438292
bruises_f                     0.369596    -0.369596      0.438292   1.000000
```

Artk görselletirme ksmna geçebiliriz. Çizdirdiimiz görselin büyüklüü ve çözünürlüünü deitirmek adna matplotlib kütüphanesini içeri aktaryorum. figsize ile boyut, dpi ile çözünürülük ayarlanabilmektedir. heatmap içindeki "annot" ile karelerin içerisine deerlerini yazdrabiliyorum.

```
In [130]: import matplotlib.pyplot as plt

          plt.figure(figsize=(10, 10), dpi=400)
          sns.heatmap(X_encoded[X_reduced_col_names].corr().abs(), annot=True)

Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x230f59fe860>
```



6 Teekkürler!