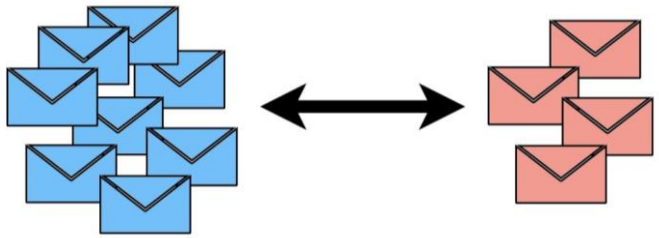
DATA MINING
PRESENTATION

HOMEWORK- PROJECT

Naive Bayes....



Uygulama adımları:

Eksik değerler
Sapan değerler
Kutulama
Çevirme
Yeni değişken üretme

Naive bayes algoritmayı kullanarak sınıflandırma model kurmak

Veri sert eğitim ve teste ayırdıktan sonra X,Y e ayırıp eğitim verileriyle model eğitimek

Test verileriyle model sınamak ve çıkan sonuçları değerlendirmek (birden farklı doğrulama yöntemi kullanarak)

Projenin skorları

Sensitivity **72,3300%**

Specificity **42,8571%**

PROJE ANLATIMI

PROJE ŞU TEKİKLERLE UYGULANDI:

PROGRAMLAMA DİLİ – PYTHON
VERİ SETİ TEST VE TRAIN - AYRI CSV DOSYALAR
FRAMEWORK – PyCHARM COMMUNITY

...

Projede gerekli kütüphaneler

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Qt5Agg')
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
```

1

Veri analizi

- Veri okumak

kod

```
df_train = pd.read_csv("datasets/trainSet.csv")
df_test = pd.read_csv("datasets/testSet.csv")
```

Pandas kütüphaneden read_csv fonksyonu kullanarak veri setleri okuyup değişkenlere atadım

- Veri setleri genel bilgilerine bakmak:

kod

```
def control(df):
    print("general information for set")
    print("##############shape################")
    print(df.shape)
    print("##############type################")
    print(df.dtypes)
    print("##############discribe################")
    print(df.describe().T)
    print("################null value count#############")
    print(df.isnull().sum())
    print("##############unique values count################")
    print(df.nunique())
    print("###############Head###############")
    print(df.head())
    print("###############Taill###############")
    print(df.tail())
```

```
# general information for train set
print("general information for train set")
control(df_train)
# general information for test set
print("general information for test set")
control(df_test)
```

- Veri setleri genel bilgilere bakmak- devamı:

sonuç

```
##############shape#################
(750, 6)
##############type#################
credit_history        object
credit_amount         object
employment            object
property_magnitude    object
age                   object
class                 object
dtype: object
##############discribe#################
                  count unique              top freq
credit_history      750      6  'existing paid'  394
credit_amount       750    708             1258    3
employment          750      5           1<=X<4  255
property_magnitude  750      5              car  253
age                 750     54               27   39
class               750      2             good  524
##################null value count##############
```

```
##################null value count##############
credit_history        0
credit_amount         0
employment            0
property_magnitude    0
age                   0
class                 0
dtype: int64
##############unique values count#################
credit_history          6
credit_amount         708
employment              5
property_magnitude      5
age                    54
class                   2
dtype: int64
```

- Veri setleri genel bilgilere bakmak- devamı:

Sonuç2

```
################Head################
                   credit_history credit_amount employment   property_magnitude age class
0                 'existing paid'          1924     1<=X<4     'life insurance'  38  good
1                 'existing paid'          7297        >=7  'no known property'  36   bad
2                 'existing paid'          1278        >=7        'real estate'  36  good
3                 'existing paid'          2039     1<=X<4        'real estate'  20   bad
4  'critical/other existing credit'          4272        >=7     'life insurance'  24  good
################Taill################
                     credit_history credit_amount employment property_magnitude age class
745                 'existing paid'          2577     1<=X<4                car  42  good
746                 'existing paid'           804        >=7                car   ?  good
747  'critical/other existing credit'          2058     1<=X<4      'real estate'  33  good
748  'critical/other existing credit'          5842        >=7   'life insurance'  35  good
749                 'existing paid'          1007     1<=X<4      'real estate'  22  good
```

- Değişkenler kategorik ve sayısala ayırmak

kod

```
# ######################analysis of variable's type############################
num_list = [col for col in df_train.columns if df_train[col].nunique() > 10]
cat_list = [col for col in df_train.columns if df_train[col].nunique() <= 10]
```

Verilerin genel bilgilerine baktıktan sonra her değişkene sayısala veya kategoriğe ayırdım.
Kullanıldığı şart her zaman değişkenleri ayırmak için geçerli değil , veri setlere bakarak yaptım hem de değişkenler az olduğu için takip etmek kolaydır

sonuç

```
In [7]: num_list
Out[7]: ['credit_amount', 'age']
In [8]: cat_list
Out[8]: ['credit_history', 'employment', 'property_magnitude', 'class']
```

- Kategorik değişkenleri özet analizi:

kod

```
# ##########################categorical variable summery#######################
for col in cat_list:
    print(pd.DataFrame({col: df_train[col].value_counts(),
                        "rate": 100 * df_train[col].value_counts() / len(df_train)}))
```

Her kategorik değişkenin unique değerlere göre değer sayısı ve yüzde oranı

sonuç

```
                                credit_history     rate
'existing paid'                            394 52.53333
'critical/other existing credit'           218 29.06667
'delayed previously'                        62  8.26667
'all paid'                                  40  5.33333
'no credits/all paid'                       31  4.13333
?                                            5  0.66667
           employment     rate
1<=X<4            255 34.00000
>=7               187 24.93333
4<=X<7            129 17.20000
<1                129 17.20000
unemployed         50  6.66667
```

```
                    property_magnitude     rate
car                                253 33.73333
'real estate'                      201 26.80000
'life insurance'                   173 23.06667
'no known property'                120 16.00000
?                                    3  0.40000
      class     rate
good    524 69.86667
bad     226 30.13333
```

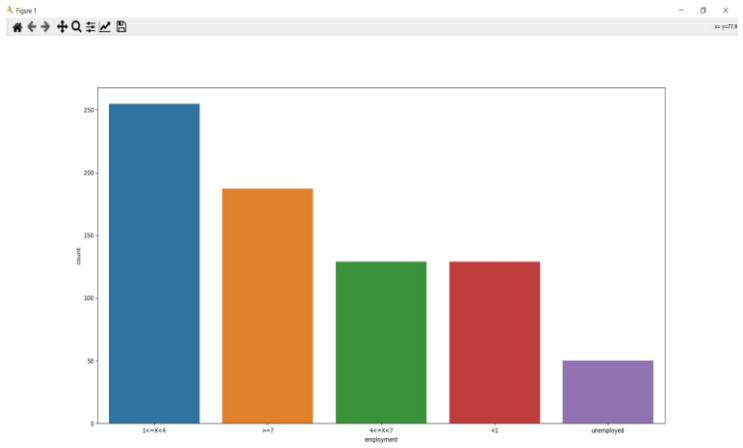
- Kategorik değişkenleri özet analizi- Grafiklar:

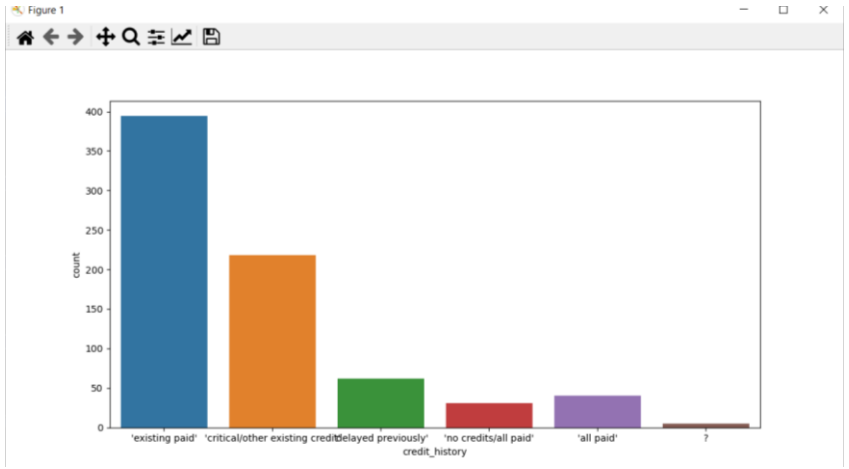
kod

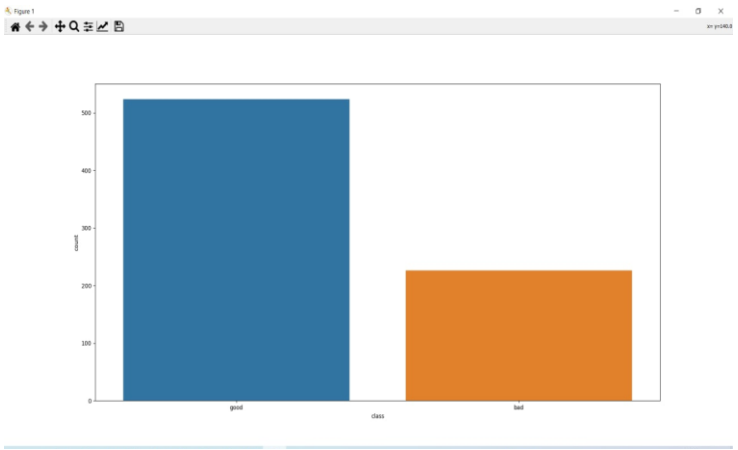
sonuç

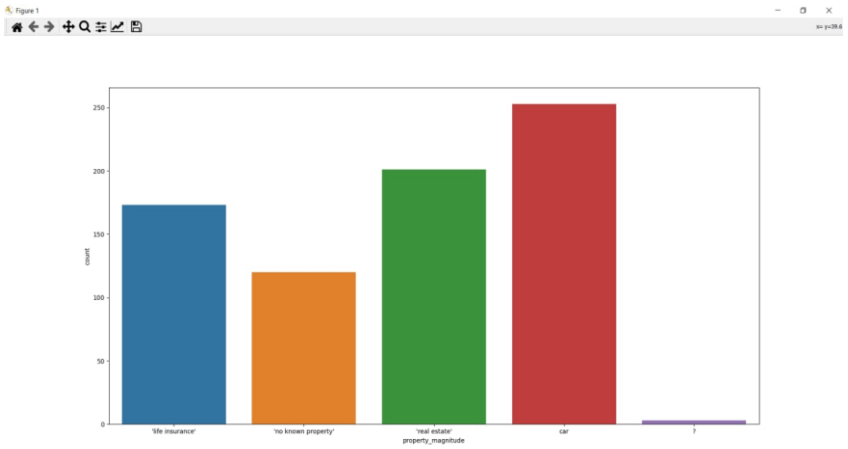
Histogram Grafikleri kullanarak kategorik değişkenleri analizi etmek

```
# ########################categorical variable graf###
sns.countplot(x=df_train[cat_list[0]], data=df_train)
plt.show(block=True)

sns.countplot(x=df_train[cat_list[1]], data=df_train)
plt.show(block=True)

sns.countplot(x=df_train[cat_list[2]], data=df_train)
plt.show(block=True)

sns.countplot(x=df_train[cat_list[3]], data=df_train)
plt.show(block=True)
```









- Kategorik değişkenleri hedef değişkene göre özet analizi:

kod

```
# ###################categorical variable summery with class(output variable)###########
for col in cat_list:
    print(pd.DataFrame({"target_count": df_train.groupby(col)["class"].value_counts()}), end="\n\n\n")
```

Kategorik değişkenler class değişkene göre analizi etmek (unique değerler sayısı ve yüzde oranı ile)

sonuç

		target_count
credit_history	class	
'all paid'	bad	26
	good	14
'critical/other existing credit'	good	183
	bad	35
'delayed previously'	good	43
	bad	19
'existing paid'	good	270
	bad	124
'no credits/all paid'	bad	20
	good	11
?	good	3
	bad	2

		target_count
employment	class	
1<=X<4	good	185
	bad	70
4<=X<7	good	96
	bad	33
<1	good	72
	bad	57
>=7	good	140
	bad	47
unemployed	good	31
	bad	19

		target_count
property_magnitude	class	
'life insurance'	good	117
	bad	56
'no known property'	good	66
	bad	54
'real estate'	good	163
	bad	38
?	good	2
	bad	1
car	good	176
	bad	77

- Sayısal değişkenleri özet analizi:

kod

```
# ######################numerical variable summery for train set#########################
for col in num_list:
    df_train[col].describe().T
```

sonuç

Credit amount değişkeni

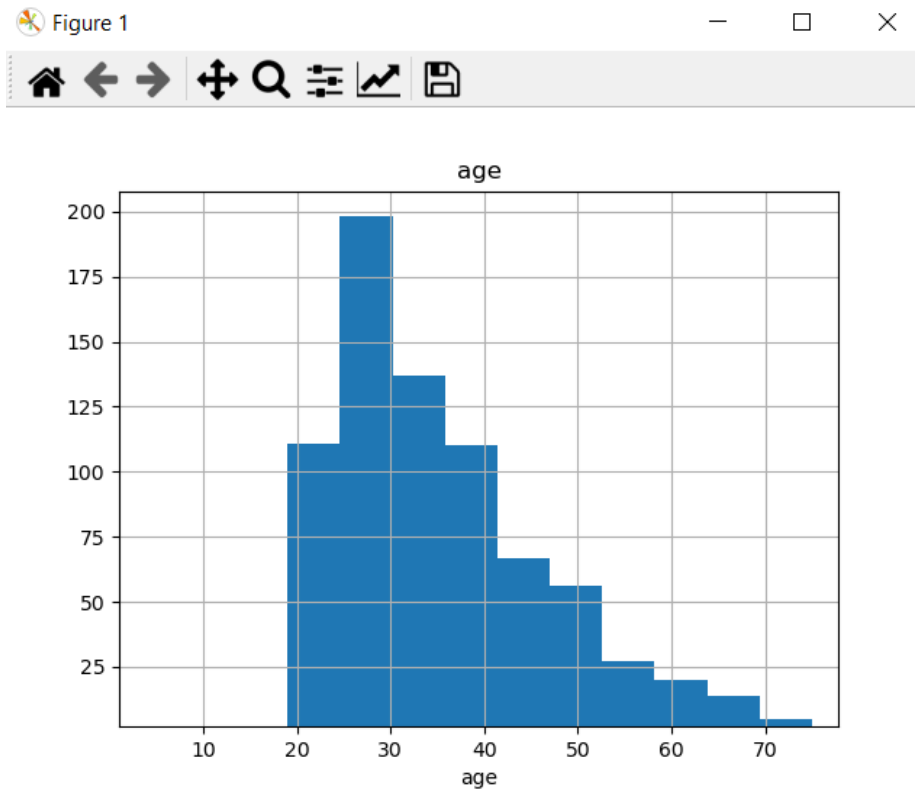
count	745.00000
mean	3258.65464
std	2846.04887
min	250.00000
25%	1347.00000
50%	2301.00000
75%	3931.00000
max	18424.00000

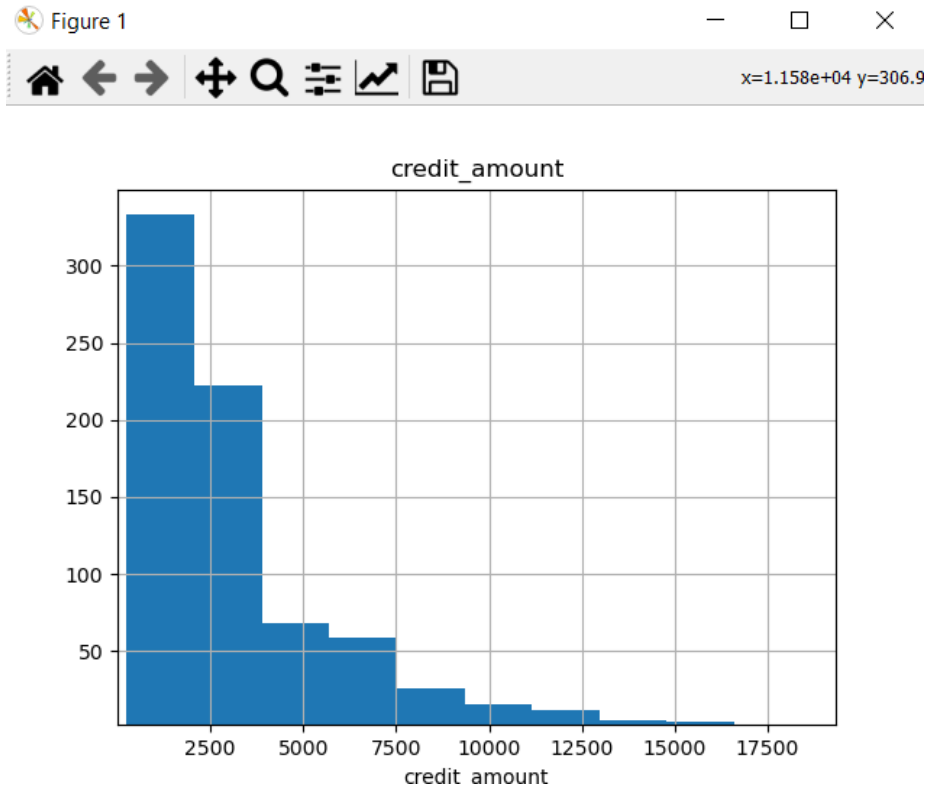
Sapan değer fark edebiliriz

Age değişkeni

count	745.00000
mean	35.37072
std	11.24380
min	19.00000
25%	27.00000
50%	33.00000
75%	42.00000
max	75.00000

- Sayısal değişkenleri özet analizi- Grafiklar:





- Sayısal değişkenleri hedef değişkene göre özet analizi:

kod

```
# ####################numerical variable summery with class(output variable)############
for col in num_list:
    print(df_train.groupby("class").agg({col: "mean"}), end="\n\n\n")
```

sayısal değişkenlerin ortalaması,max ve min değerleri class değişkene göre analizi etmek

sonuç

```
                mean          max         min
class
bad       3991.35268 18424.00000 433.00000
good      2943.63667 15653.00000 250.00000



           age
          mean       max       min
class
bad    33.55357 74.00000 20.00000
good   36.15198 75.00000 19.00000
```

2

Veri ön işlemi

- Eksik değerler

Veri setlerinde eksik değerler ? İşaretle doldurulmuş.

Fazla veri kayıbı olmaması için şu adımlar uyguladık:

1- train veri setindeki sayısal değişkenler için ortalama ile doldurmak.(age – credit_amount), test veri ise kayıtları silmek.

2- property_magnitude değişken için ? Yerine “no known property” ile doldurmak çünkü zaten öyle bir değerler var.

3- diğer kategorik değişkenler için eksik değerleri olan kayıdı silmek

- Eksik değerler – devamı

kod

```
# ################################check missing values#####################################
# for train set
for col in df_train.columns:
    print(col, df_train[df_train[col] == '?'].any().sum())

# for train set
for col in df_test.columns:
    print(col, df_test[df_test[col] == '?'].any().sum())
```

sonuç

Eksik değerleri sayısına bakmak

```
credit_history 6
credit_amount 6
employment 0
property_magnitude 6
age 6
class 0
```

```
credit_history 6
credit_amount 6
employment 6
property_magnitude 6
age 6
class 0
```

- Eksik değerler – devamı

kod

```
# ############################## missing values solution#################################
# solution num variable for train set (Replace the missing values with the arithmetic mean)
for col in num_list:
    df_train.loc[(df_train[col] == '?'), col] = 0
    # from string to float
    df_train[col] = df_train[col].astype(float)
    # from 0 to mean
    df_train.loc[(df_train[col] == 0), col] = df_train[col].mean()
```

Train set

Not1: ortalama hesaplamak için sütün tipi flaota dönüştümemiz gerekti

Sayısal değişkenler için eksik değerler yerine aritmetik ortalama(değişkenin değerleri ortalaması) ile doldurmak
Test veri seti için sildik

Not2: önce ? Yerine 0 atadık, hem tip değiştirebilmek için hem ortalama hesaplamak için

```
df_test = df_test[~(df_test["age"] == '?')]
df_test = df_test[~(df_test["credit_amount"] == '?')]
```

Test set

- Eksik değerler – devamı

kod

```
# solution cat variable for train set (Delete entries that contain values?)
df_train = df_train[~(df_train["credit_history"] == '?')]
df_train.loc[(df_train["property_magnitude"] == '?'), "property_magnitude"] = 'no known property'
```

Train set

Kategorik değişkenlerin eksik değerleri silme kodları

```
df_test.loc[(df_test["property_magnitude"] == '?'), "property_magnitude"] = 'no known property'
df_test = df_test[~(df_test["credit_history"] == '?')]
df_test = df_test[~(df_test["employment"] == '?')]
```

Test set

- Sapan değerler (outlier values)

kod

Grafikları kullanarak sapan değerleri olup olmadığına kontrol ediyoruz

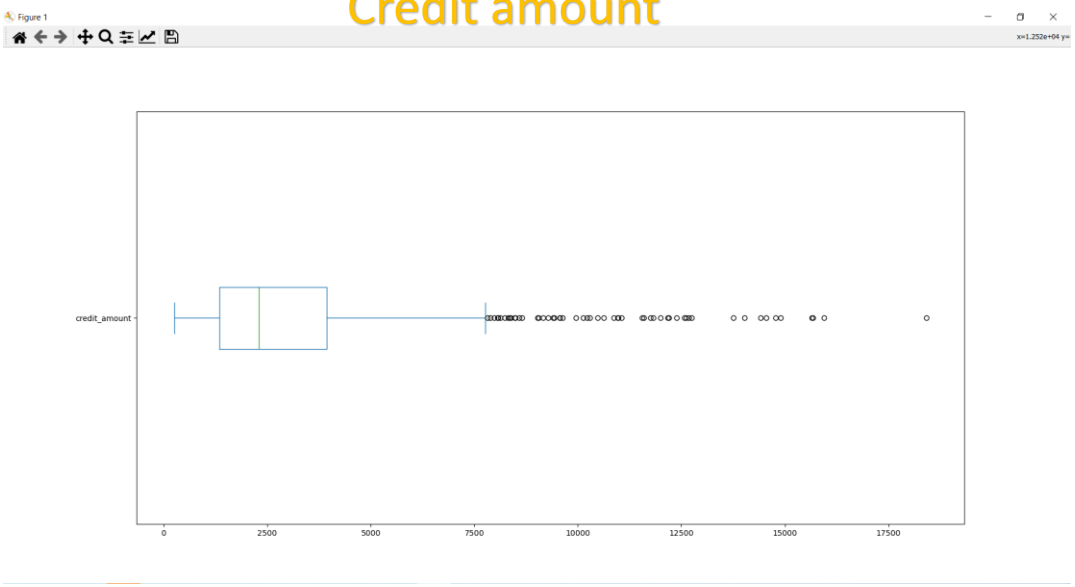
```
# ############################See outlier values with graphs#################################
# for data train
 df_train["age"].plot(kind="box", vert=False)
df_train["credit_amount"].plot(kind="box", vert=False)
```

- Sapan değerler (outlier values)- devamı:

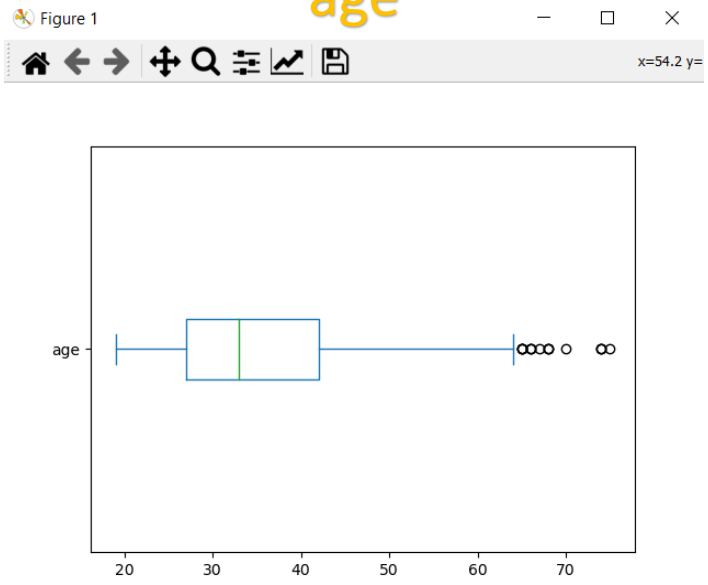
Kutu grafikları kullanarak sapan değerleri olup olmadığına kontrol ediyoruz

sonuç

Credit amount



age



Grafiklere bakarak credit amount değişkeninde fark edilir bir sapan değer gözüküyor

Projede sapan değerleri silmek veya düzeltmeyi tercih etmedim

- Veri dönüştürmek

Veri setinde 3 çoklu değer ve ordinal olmayan kategorik değişken var ve ikili değer kategorik değişken var o da class hedef değişkenimiz.

Veri dönüştürmek için:

1- çoklu değerler için one hat encoding bir yöntem kullanacağız (dummy hazır fonksiyounla)

2- class değişken için labelencoding (binary encoding) yöntemi kullanacağız.

- Veri dönüştürmek

- One hat encoding

kod

```
# ##########################One hat encoding for train set#########################################
cat_list1 = [col for col in df_train.columns if (df_train[col].nunique() <= 10) and (df_train[col].nunique() > 2)]
# for train data
df_train_prep3 = df_train
for col in cat_list1:
    dummies = pd.get_dummies(df_train_prep3[col])
    df_train_prep3 = pd.concat([df_train_prep3, dummies], axis="columns")
    df_train_prep3.drop([col], axis="columns", inplace=True)
df_train_prep3.head()
```

Eski sütünü siliyoruz

```
# for test data
df_test_prep4 = df_test
df_test_prep4.head()
for col in cat_list1:
    dummies = pd.get_dummies(df_test_prep4[col])
    df_test_prep4 = pd.concat([df_test_prep4, dummies], axis="columns")
    df_test_prep4.drop([col], axis="columns", inplace=True)
df_test_prep4.head()
```

- Veri dönüştürmek

- One hat encoding

sonuç

	credit_amount	age	class	'all paid'	'critical/other existing credit'	'delayed previously'	'existing paid'	'no credits/all paid'	1<=X<4	4<=X<7	<1	>=7	unemployed	'life insurance'	'no known property'	'real estate'	car	no known property
0	10366.00000	42.00000	good	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0
1	1872.00000	36.00000	good	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0
2	6758.00000	31.00000	bad	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0
3	3857.00000	40.00000	good	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0
4	3190.00000	24.00000	bad	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0

- Veri dönüştürmek

  - İkili değer değişken veri dönüştürme için

kod

```
# ####################################binary encoding for output variable for train set####
coder = LabelEncoder()
df_train_prep3["class"] = coder.fit_transform(df_train_prep3["class"])
df_train_prep3["class"].value_counts()
# ####################################binary encoding for output variable for test set#####
coder = LabelEncoder()
df_test_prep4["class"] = coder.fit_transform(df_test_prep4["class"])
df_test_prep4["class"].value_counts()
```

Train set

Test set

- Veri dönüştürmek

- İkili değer değişken veri dönüştürme için

sonuç

```
df_train_prep3["class"].head()
```

```
In [34]: df_train_prep3["class"].head()
Out[34]:
0    1
1    0
2    1
3    0
4    1
Name: class, dtype: int32
```

- Veri kutulama (binning)

- Veri setindeki sayısal değişkenler için kutulama yapabiliriz Equal width yöntemi kullanabiliriz

<u>Önemli notlar:</u>
1- hem credit amount için hem de age için kutulama yaptım
2- çıkan sonuçlar yeni bir sütüne atadım
3- kutulama yaptıktan sonra sayısal sütünlari silerek ve silmeden
İki yöntemle deneyerek veri seti modele verdim ancak doğruluk değerleri Yükselmeyip tersine daha düşük çıktığı için binning işleminde vazgeçtim

Kodları sadece göz atmak için ekliyorum.

- Veri kutulama (binning)

kod

```
# ####################Binning(equal width binning) to  variable "credit_amount" into 4 categories############
df_train["credit_level"] = pd.cut(df_train["credit_amount"], bins=4, labels=["low", "average", "high", "very high"])
df_test["credit_level"] = pd.cut(df_test["credit_amount"], bins=4, labels=["low", "average", "high", "very high"])
print(df_train.groupby("credit_level").agg({"credit_amount": ["mean", "count", "max", "min"]}), end="\n\n\n")

# ####################Binning(equal width binning) to  variable "age" into 4 categories###############
df_train["age-level"] = pd.cut(df_train["age"], bins=4, labels=["young", "young man", "senior", "super senior"])
df_test["age-level"] = pd.cut(df_test["age"], bins=4, labels=["young", "young man", "senior", "super senior"])
print(df_train.groupby("age-level").agg({"age": ["mean", "count", "max", "min"]}), end="\n\n\n")
```

3

Model kurma

- Naive bayes – model kurmak

kod

Hazır kütüphaneden naive bayes algoritmalı modeli kurdum
Kod şu şekildedir:

```
from sklearn.naive_bayes import GaussianNB
```

```
model2 = GaussianNB()
```

4

Model eğitme

- Model eğitmek

Model kurduktan sonra eğitim veri seti hem x hem y şeklinde ayırdım
X = bütün değişkenler class hedef değişkenimiz hariç.
Y = class hedef değişkenimiz.
Ve fit fonksiyonu kullanarak modeli eğittim

kod

```
# ########################################model#####################################
y_train = df_train_prep3["class"]
x_train = df_train_prep3.drop("class", axis="columns")
y_test = df_test_prep4["class"]
x_test = df_test_prep4.drop("class", axis="columns")

model2 = GaussianNB()
model2.fit(x_train, y_train)
```

5

Model doğrulama

- Doğrulama fonksiyonu kodlamak:

- Python dilinde gibi hazır doğrulama ölçme fonksiyonları var ancak projenin isteklerine göre sıfırdan kodlayacağız
Ve fonksiyon şu şekilde olacak:

kod

```
# ########################################Ac
def evaluation(y, y_pre):
    tp = sum((y == 1) & (y_pre == 1))
    tn = sum((y == 0) & (y_pre == 0))
    fp = sum((y == 0) & (y_pre == 1))
    fn = sum((y == 1) & (y_pre == 0))
    tp_r = (tp/(tp+fp))
    tn_r = (tn/(tn+fn))
    acc = ((tp+tn)/(tp+fp+tn+fn))
    print("test results is")
    print("acc is =", acc)
    print("true positive count is =", tp)
    print("true positive rate is =", tp_r)
    print("true negative count is =", tn)
    print("true negative rate is =", tn_r)
```

Fonksiyon şu şekilde çalışır:

1- iki parametre alır (modelin tahmin ettiği değerler, gerçek değerler)

2- iki parametre karşılaştıracak ve sonuç olarak şu değerler return edecek:

1- accuracy

2- true positive sayısı

3- true negative sayısı

4- Sensitivity

5- Specificity


skor

- Modelin doğruluk sonuçları:

kod

Modelin
Test x aldıktan
sonra tahmin
ettiği sınıf değeri

```
y_predicted = model2.predict(x_test)

evaluation(y_test, y_predicted)
```

Doğruluk
fonksiyonu
çağırmak

sonuç

```
test results is
acc is = 0.6804979253112033
true positive count is = 149
true positive rate is = 0.7233009708737864
true negative count is = 15
true negative rate is = 0.42857142857142855
```