KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

MAKİNE ÖĞRENMESİ

KREDİ DURUM TAHMİNİ

RAMAZAN ÖZER — **18110131027**

MERVE ÇÜRÜK — **18110131312**

Dr.Öğr.Üyesi YAVUZ CANBAY

ÖZET

Makine öğrenmesi, insanların öğrenme şekillerini taklit etmek için veri ve algoritmaların kullanımına odaklanıp doğruluğunu kademeli olarak artıran bir yapay zeka (AI) ve bilgisayar bilimi dalıdır. Sınıflandırma için Destek Vektör Makine Öğrenmesi, KNN (En Yakın Komşuluk) Algoritması ve Rastgele Orman Algoritması kullandık.

GİRİŞ

Bu çalışmada kişilerin cinsiyet, evlilik, eğitim durumları, gelir, kredi miktarlarına vb. çeşitli özelliklere bakılarak sınıflandırma algoritmaları ile kredi alıp almamalarını hesaplayan makine öğrenmesi uygulaması yaptık.

Problemin Tanımı/Konunun Tanımı

Problemimizin 13 değişkenli olan bir verimizden değişkenler arasında nasıl bir uyum ve bu uyumu kredi durumu ile alakalı olduğunu bulmamız ve makine öğrenmesi ile modelleme işlemi yapmamız gerekiyor. Çeşitli türlerde bu işlemler farklı şekilde yapılarak farklı sonuçlar alabiliyoruz.

İçindekiler

SİSTEM TASARIMI

Yukarıdaki maddelere göre projemizi tasarladık.

VERİ TOPLAMA

VERİ SETİ HAKKINDA ÖN BİLGİ

Veri setimizi kaggle sitesinden (Link= https://www.kaggle.com/ninzaami/loan-predication) bulduk.

```
# Veri kümesini pandaların DataFrame'ine yükleme
loan_dataset = pd.read_csv('veri.csv')
```

Verimizdeki ilk 5 satırı, satır ve sütun sayılarını yazdırdık.

```
loan_dataset.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0

```
# Satır ve Sütun sayısı
loan_dataset.shape
```

```
(614, 13)
```

Veride her sütundaki eksik değer sayısını öğrendik.

```
loan_dataset.isnull().sum()
```

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

VERİ ÖN İŞLEME

Değişkenlerimizi sayısallaştırdık. Ayrıca 3 tane ayrı veri oluşturduk.

```
# Etiket kodlaması
loan_dataset.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
```

```
# Veri çerçevesinin ilk 5 satırını yazdırma
loan_dataset.head()
```

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	1
Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	0
Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	1
Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	1
No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	1

Dependents kolonundaki 3+ string değerini 4 integer değere dönüştürdük.

```
In [10]: # Bağımlı sütun değerleri
         loan_dataset['Dependents'].value_counts()
```

```
Out[10]: 0     345
         1     102
         2     101
         3+     51
         Name: Dependents, dtype: int64
```

```
In [11]: # 3+ değerinin 4 ile değiştirilmesi
         loan_dataset = loan_dataset.replace(to_replace='3+', value=4)
```

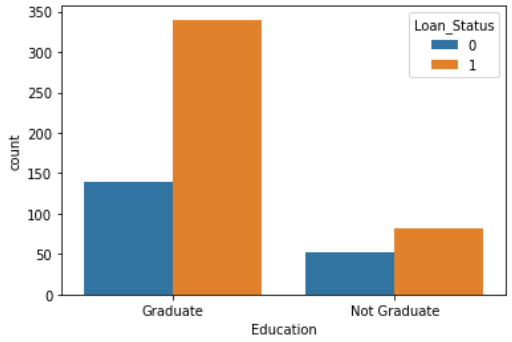
```
In [12]: # Bağımlı değerler
         loan_dataset['Dependents'].value_counts()
```

```
Out[12]: 0    345
         1    102
         2    101
         4     51
         Name: Dependents, dtype: int64
```

Eğitim ve kredi durumunu grafik ile oluşturduk.

```
In [13]: # Eğitim ve Kredi Durumu
         sns.countplot(x='Education',hue='Loan_Status',data=loan_dataset)
```

```
Out[13]: <AxesSubplot:xlabel='Education', ylabel='count'>
```




```
In [16]: # Kategorik sütunları sayısal değerlere dönüştürdük.
         loan_dataset.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self_Employed':{'No':0,'Yes':1},
                               'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{'Graduate':1,'Not Graduate':0}},inplace=T
```

```
In [ ]: # Kişilerin Id sütunu işimize yaramadığından veriden kaldırıyoruz.
        loan_dataset.drop(columns=['Loan_ID'],axis=1,inplace=True)
```

```
In [18]: # Her sütundaki eksik değer sayısı
         loan_dataset.isnull().sum()
```

```
Out[18]: Gender              13
         Married              3
         Dependents          15
         Education            0
         Self_Employed       32
         ApplicantIncome      0
         CoapplicantIncome    0
         LoanAmount          22
         Loan_Amount_Term    14
         Credit_History      50
         Property_Area        0
         Loan_Status          0
         dtype: int64
```

Eksik verilerin düşülmesi ile oluşan veri.

```
In [42]: # Eksik değerlerin düşürülmesi ile oluşan veri
         Data_Copy=loan_dataset.copy(deep=True)
         Data_Copy = loan_dataset.dropna()
```

```
In [43]: # Her sütundaki eksik değer sayısı
         Data_Copy.isnull().sum()
```

```
Out[43]: Gender              0
         Married             0
         Dependents          0
         Education           0
         Self_Employed       0
         ApplicantIncome     0
         CoapplicantIncome   0
         LoanAmount          0
         Loan_Amount_Term    0
         Credit_History      0
         Property_Area       0
         Loan_Status         0
         dtype: int64
```

Verileri Mod İle Doldurma

Orijinal veriyi tutup doldurma işlemleri için Mode ()[0]'ı kullandık.

```
Mod_Data=loan_dataset.copy(deep=True)

Mod_Data['Gender']=Mod_Data['Gender'].fillna(Mod_Data['Gender'].mode()[0])
Mod_Data['Married']=Mod_Data['Married'].fillna(Mod_Data['Married'].mode()[0])
Mod_Data['Dependents']=Mod_Data['Dependents'].fillna(Mod_Data['Dependents'].mode()[0])
Mod_Data['Self_Employed']=Mod_Data['Self_Employed'].fillna(Mod_Data['Self_Employed'].mode()[0])
Mod_Data['LoanAmount']=Mod_Data['LoanAmount'].fillna(Mod_Data['LoanAmount'].mode()[0])
Mod_Data['Loan_Amount_Term']=Mod_Data['Loan_Amount_Term'].fillna(Mod_Data['Loan_Amount_Term'].mode()[0])
Mod_Data['Credit_History']=Mod_Data['Credit_History'].fillna(Mod_Data['Credit_History'].mode()[0])
```

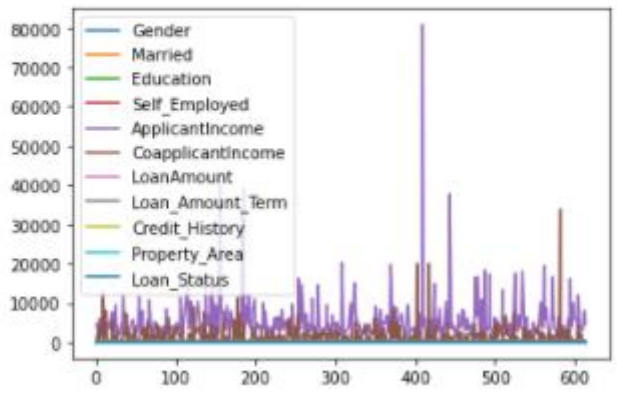
```
In [47]: # Mod_Data verisindeki her sütundaki eksik değer sayısı
         Mod_Data.isnull().sum()
```

```
Out[47]: Gender              0
         Married             0
         Dependents          0
         Education           0
         Self_Employed       0
         ApplicantIncome     0
         CoapplicantIncome   0
         LoanAmount          0
         Loan_Amount_Term    0
         Credit_History      0
         Property_Area       0
         Loan_Status         0
         dtype: int64
```

```
In [29]: Data_Copy.duplicated()#tekrarlanan verileri kontrol etmek
```

```
Out[29]: 1      False
         2      False
         3      False
         4      False
         5      False
                ...
         609    False
         610    False
         611    False
         612    False
         613    False
         Length: 480, dtype: bool
```

```
In [55]: # tüm verileri de çizdirebiliriz
         Data_Copy.plot()
         plt.show()
```



Model Oluşturma

Eğitim ve test verisi ayrımı

```
1-) Eğitim ve Test Verisi Ayrımı Data_Copy verisi ile
```

In [ ]:
```
# Verileri ve Etiketi ayırma
X = Data_Copy.drop(columns=['Loan_Status'],axis=1)#bağımlı değişkenler
Y = Data_Copy['Loan_Status'] #bağımsız değişkenler
```

In [ ]:
```
print(X)
print(Y)
```

In [ ]:
```
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

In [ ]:
```
print(X.shape, X_train.shape, X_test.shape)
```

DESTEK VEKTÖR MAKİNE ÖĞRENMESİ

Destek Vektör Makineleri (Support Vector Machine) genellikle sınıflandırma problemlerinde kullanılan gözetimli öğrenme yöntemlerinden biridir. Bir düzlem üzerine yerleştirilmiş noktaları ayırmak için bir doğru çizer. Bu doğrunun, iki sınıfının noktaları için de maksimum uzaklıkta olmasını amaçlar.

Modeli eğitmek:

Destek Vektör Makinesi Modeli

1-) Eğitim ve Test Verisi Ayrımı Data_Copy verisi ile

In [146]:
```
# Verileri ve Etiketi ayırma
X = Data_Copy.drop(columns=['Loan_Status'],axis=1)#bağımlı değişkenler
Y = Data_Copy['Loan_Status'] #bağımsız değişkenler
```

In [147]:
```
print(X)
print(Y)
```

```
     Gender  Married Dependents  Education  Self_Employed  ApplicantIncome  \
1       1.0      1.0          1          1            0.0             4583
2       1.0      1.0          0          1            1.0             3000
3       1.0      1.0          0          0            0.0             2583
4       1.0      0.0          0          1            0.0             6000
5       1.0      1.0          2          1            1.0             5417
..      ...      ...        ...        ...            ...              ...
609     0.0      0.0          0          1            0.0             2900
610     1.0      1.0          4          1            0.0             4106
611     1.0      1.0          1          1            0.0             8072
612     1.0      1.0          2          1            0.0             7583
613     0.0      0.0          0          1            1.0             4583

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
1               1508.0       128.0             360.0             1.0
2                  0.0        66.0             360.0             1.0
3               2358.0       120.0             360.0             1.0
4                  0.0       141.0             360.0             1.0
5               4196.0       267.0             360.0             1.0
```

```
In [148]: X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)

In [149]: print(X.shape, X_train.shape, X_test.shape)

          (480, 11) (432, 11) (48, 11)

In [150]: classifier = svm.SVC(kernel='linear')

In [151]: #Destek Vektör Makine modelini eğitmek
          classifier.fit(X_train,Y_train)

Out[151]: SVC(kernel='linear')
```

Model Değerlendirmesi

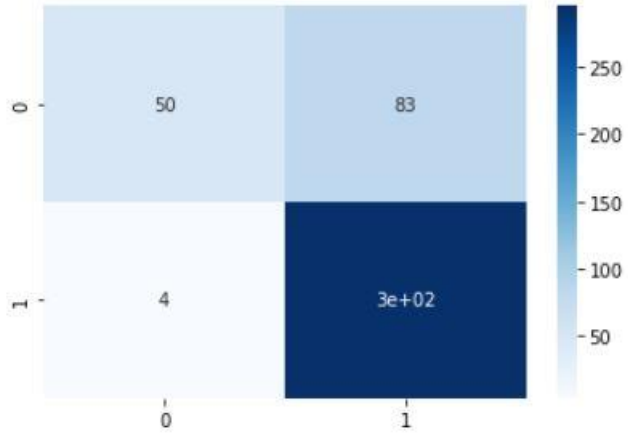
```
In [152]: # Eğitim verilerinde doğruluk puanı
          X_train_prediction = classifier.predict(X_train)
          training_data_accuray = accuracy_score(X_train_prediction,Y_train)

In [153]: print('Eğitim verilerinin doğruluğu : ', training_data_accuray)

          Eğitim verilerinin doğruluğu :  0.7986111111111112

In [154]: cf_matrix =confusion_matrix(Y_train, X_train_prediction)
          sns.heatmap(cf_matrix, annot=True,cmap='Blues')

Out[154]: <AxesSubplot:>
```



In [156]:
```
# Test verilerinde doğruluk puanı
X_test_prediction = classifier.predict(X_test)
test_data_accuray = accuracy_score(X_test_prediction,Y_test)
```

In [157]:
```
print(' Test verilerinin doğruluk :',test_data_accuray)
```

```
 Test verilerinin doğruluk : 0.8333333333333334
```

In [158]:
```
cf_matrix =confusion_matrix(Y_test, X_test_prediction)
sns.heatmap(cf_matrix, annot=True)
```

Out[158]: <AxesSubplot:>

2-) Eğitim ve Test Verisi Ayrımı Mod verisi ile

In [159]:
```
# Verileri ve Etiketi ayırma
X = Mod_Data.drop(columns=['Loan_Status'],axis=1)#bağımlı değişkenler
Y = Mod_Data['Loan_Status'] #bağımsız değişkenler
```

In [160]:
```
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

In [161]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(614, 11) (552, 11) (62, 11)
```

In [162]:
```
classifier = svm.SVC(kernel='linear')
#Destek Vektör Makine modelini eğitmek
classifier.fit(X_train,Y_train)
```

Out[162]:
```
SVC(kernel='linear')
```

Model Değerlendirmesi

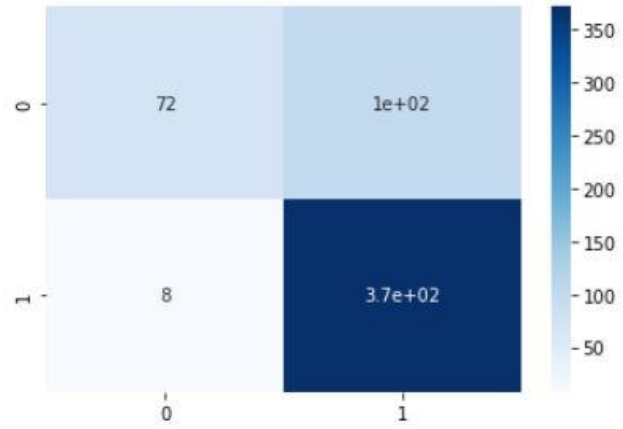
In [163]:
```
# Eğitim verilerinde doğruluk puanı
X_train_prediction = classifier.predict(X_train)
training_data_accuray = accuracy_score(X_train_prediction,Y_train)
```

In [164]:
```
print('Eğitim verilerinin doğruluğu : ', training_data_accuray)
```

```
Eğitim verilerinin doğruluğu :  0.802536231884058
```

In [165]:
```
cf_matrix =confusion_matrix(Y_train, X_train_prediction)
sns.heatmap(cf_matrix, annot=True,cmap='Blues')
```

Out[165]: <AxesSubplot:>



In [166]:
```
# Test verilerinde doğruluk puanı
X_test_prediction = classifier.predict(X_test)
test_data_accuray = accuracy_score(X_test_prediction,Y_test)
```

In [167]:
```
print(' Test verilerinde doğruluğu :',test_data_accuray)
```

```
 Test verilerinde doğruluğu : 0.8064516129032258
```

In [168]:
```
cf_matrix =confusion_matrix(Y_test, X_test_prediction)
sns.heatmap(cf_matrix, annot=True)
```

Out[168]: <AxesSubplot:>

KNN SINIFLANDIRMA

K-NN sınıflandırmasında , çıktı sınıf üyeliğidir. Bir nesne, komşularının çoğunluk oyuyla sınıflandırılır; nesne, en yakın komşuları arasında en yaygın olan sınıfa verilir (k , küçük bir pozitif bir tam sayı). Eğer k = 1 ise, nesne basitçe o en yakın komşunun sınıfına atanır.

Eğitim ve Test Verisi Ayrımı DataCopy verisi ile

```
In [169]: # Verileri ve Etiketi ayırma
          X = Data_Copy.drop(columns=['Loan_Status'],axis=1)#bağımlı değişkenler
          Y = Data_Copy['Loan_Status'] #bağımsız değişkenler
```

```
In [170]: X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)
```

```
In [173]: print(' Test verilerinde doğruluğu :',test_data_accuray)

          Test verilerinde doğruluğu : 0.6666666666666666
```

```
In [174]: cf_matrix =confusion_matrix(Y_test, Y_tahmin)
          sns.heatmap(cf_matrix, annot=True)

Out[174]: <AxesSubplot:>
```

RASTGELE ORMAN ALGORİTMASI

Random forest, birden fazla karar ağacını kullanarak daha uyumlu modeller üreterek daha isabetli sınıflandırma yapmaya çalışan bir sınıflandırma modelidir.

Eğitim ve Test Verisi Ayrımı DataCopy verisi ile

```
In [181]: # Verileri ve Etiketi ayırma
          X = Data_Copy.drop(columns=['Loan_Status'],axis=1)#bağımlı değişkenler
          Y = Data_Copy['Loan_Status'] #bağımsız değişkenler

In [182]: X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=2)

In [183]: modela = RandomForestClassifier()

In [184]: modela.fit(X_train, Y_train)
Out[184]: RandomForestClassifier()
```

Test verilerindeki doğruluk puanını bulduk.

```
          X_test_prediction = modela.predict(X_test)
          test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [186]: print(' Test verilerinin doğruluk :',test_data_accuray)

          Test verilerinin doğruluk : 0.6451612903225806

In [187]: cf_matrix =confusion_matrix(Y_test, X_test_prediction)
          sns.heatmap(cf_matrix, annot=True)
Out[187]: <AxesSubplot:>
```