BÖLÜM IV
BULGULAR VE YORUM

4.1 Uygulama

Bu uygulamada veri madenciliği algoritmaları olan Destek Vektör Makineleri ve Lojistik Regresyon ile veri setinin modellenmesi amaçlanmıştır. Kullanılan veri seti, algoritma ile eğitilmiştir. Böylece gelecek olan test veri setlerinin hasta olup olmama olasılığı araştırılmıştır. Uygulamada kullanılan veri seti diyabet hastalığı ile ilgilidir. İnternet ortamından excel dosya formatı olan '.csv' uzantısı olarak indirilmiştir. Kullanılan program ".csv" formatında analiz yapmayı desteklemektedir. Bu sebeple indirilen dosya analiz işlemlerinde direk kullanılmıştır. Python programlama dili kullanılarak, Anaconda Navigator içerisinde Jupyter ortamında yazılmıştır. Öncelikle Jupyter ortamına kullanılacak olan Numpy ve Pandas Python kütüphaneleri Şekil 4.1'de görüldüğü üzere yüklenmiştir.

Şekil 4.1.
Python kütüphanelerinin yüklenmesi

```
import numpy as np
import pandas as pd
```

Veriler program içerisine yüklenmiştir. Veri seti içerisinde 768 adet kişinin bilgileri bulunmaktadır. Veri setindeki her hayıt bir kişiye ait özelliklerden oluşmaktadır. Kişilerin hepsi bayandır. Outcome değeri ise çıktı değeridir. Değer 1 ise diyabet hastası olduğunu, 0 ise diyabet hastası olmadığını gösterir. Şekil 4.2'de görüldüğü üzere veri seti 8 özellik 'ten oluşmuştur. Bu özellikler şunlardır:

- Pregnancies: Kişinin kaç defa hamile kaldığı belirtilmiştir.
- Glucose: Glikoz değeridir.
- Blood Pressure: Kan basıncı değeridir.
- Skin Thickness: Cilt kalınlığı değeridir.
- İnsülin: İnsülin değeridir.
- Body Mass Index: Vücut kitle indeksi değeridir.
- Diabetes Pedigree Function: Diyabet soyağacı fonksiyonu değeridir.
- Age: Kişilerin yaşını gösteren değerdir.

Şekil 4.2.
Veri seti özellikleri

```
df = pd.read_csv("diabetes.csv")
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Daha sağlıklı sonuçların çıkması için veri ön işleme adımı önemlidir. Veri ön işleme işlemi adımı için her bir özellikte kaç adet 0 değerinin olduğu Şekil 4.3. ve Şekil 4.4'de öğrenildi.

Şekil 4.3.
Veri seti sıfır değerlerinin toplamı

```
df.eq(0).sum()
```

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

Şekil 4.4.
Veri seti değerleri

```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']]
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	148	72	35	0	33.6	0.627	50
1	85	66	29	0	26.6	0.351	31
2	183	64	0	0	23.3	0.672	32
3	89	66	23	94	28.1	0.167	21
4	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...
763	101	76	48	180	32.9	0.171	63
764	122	70	27	0	36.8	0.340	27
765	121	72	23	112	26.2	0.245	30
766	126	60	0	0	30.1	0.349	47
767	93	70	31	0	30.4	0.315	23

768 rows × 7 columns

Glikoz, kan basıncı, cilt kalınlığı, insülin, vücut kitle indeksi, diyabet soy ağacı fonksiyonu ve yaş özellikleri sütunlarında yer alan 0 değerleri NaN olarak değiştirildi. Bu hücreleri Şekil 4.5'de eksik değerler olarak tanımlandı.

Şekil 4.5.
Eksik değerlerin tanımlanması

```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']].replace(0,np.NaN)
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	148.0	72.0	35.0	NaN	33.6	0.627	50
1	85.0	66.0	29.0	NaN	26.6	0.351	31
2	183.0	64.0	NaN	NaN	23.3	0.672	32
3	89.0	66.0	23.0	94.0	28.1	0.167	21
4	137.0	40.0	35.0	168.0	43.1	2.288	33
...	...	...	...	...	...	...	...
763	101.0	76.0	48.0	180.0	32.9	0.171	63
764	122.0	70.0	27.0	NaN	36.8	0.340	27
765	121.0	72.0	23.0	112.0	26.2	0.245	30
766	126.0	60.0	NaN	NaN	30.1	0.349	47
767	93.0	70.0	31.0	NaN	30.4	0.315	23

768 rows × 7 columns

Eksik değerleri kendi sütununda bulunan değerlerin ortalaması ile Şekil 4.6'da dolduruldu. Bu sayede veri setinde eksik değer kalmadı.

Şekil 4.6.
Eksik değerlerin doldurulması

```
df.fillna(df.mean(),inplace = True)
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.00000	155.548223	33.6	0.627	50	1
1	1	85.0	66.0	29.00000	155.548223	26.6	0.351	31	0
2	8	183.0	64.0	29.15342	155.548223	23.3	0.672	32	1
3	1	89.0	66.0	23.00000	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.00000	168.000000	43.1	2.288	33	1

Eksik değer kalmadığı Şekil 4.7'de görülüyor.

Şekil 4.7.
Eksik değerlerin toplamı

```
df.eq(0).sum()
```

```
Pregnancies                 111
Glucose                       0
BloodPressure                 0
SkinThickness                 0
Insulin                       0
BMI                           0
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

Diyabet hastası olup olmadığına en çok etki eden 4 özelliği Şekil 4.8'de görüldüğü üzere korelasyon analizine göre bulundu. Bunlar sırasıyla glikoz, vücut kitle endeksi ve yaş özellikleridir. Model oluştururken bu özellikler kullanılacaktır.

Şekil 4.8.
En çok etki eden 4 özellik

```
df.corr().nlargest(4,'Outcome').index
```

```
Index(['Outcome', 'Glucose', 'BMI', 'Age'], dtype='object')
```


Veri ön işleme ve en çok etki eden özelliklerin ortaya çıkarılmasının ardından veri analize hazır hale geldi. Kullanılacak olan algoritmalar lojistik regresyon ve destek vektör regresyon 'dur. Algoritmaların oluşturulabilmesi için Python'da bulunan hazır kütüphaneler Şekil 4.9'da programa dahil edildi.

Şekil 4.9.
Algoritmaların Oluşturulması

```
# Algoritmaların Oluşturulması

from sklearn import linear_model
from sklearn.model_selection import cross_val_score
from sklearn import svm
```

X ve y değişkenleri Şekil 4.10'da belirlendi. X değişkeni özellik setini, y değişkeni ise çıktı setini temsil eder.

Şekil 4.10.
X ve y değişkenlerinin belirlenmesi

```
X = df[['Glucose','BMI','Age']]
y = df.iloc[:,8]
```

X değişkeni Şekil 4.11'de Glikoz, Vücut Kitle İndeksi ve yaş değerlerini gösterir.

Şekil 4.11.
X değişkeninin gösterilmesi

```
X
```

	Glucose	BMI	Age
0	148.0	33.6	50
1	85.0	26.6	31
2	183.0	23.3	32
3	89.0	28.1	21
4	137.0	43.1	33
...	...	...	...
763	101.0	32.9	63
764	122.0	36.8	27
765	121.0	26.2	30
766	126.0	30.1	47
767	93.0	30.4	23

768 rows × 3 columns

y değişkeni Şekil 4.12'de Outcome sütunundaki 0 ve 1 değerlerini gösterir.

Şekil 4.12.
y değişkeninin gösterilmesi

```
y
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Lojistik regresyonu oluşturup, skoru kayıt etme işlemi Şekil 4.13'de yapıldı. X ve y değişkenlerini ayırıp, her on tane de bir gözlem olarak belirlendi. Doğrulukların ortalaması alınarak skor belirlendi. Bunun anlamı on tane de bir öğrenme gerçekleştirerek "accuracy" skorunu belirleyip, ortalamasının hesaplanarak bulunmasıdır. 0.7669856450330144 değeri veri setindeki sonuçların %76 doğru şekilde bildiğini gösterir. Bu değer, başarı oranını gösterir.

Şekil 4.13.
Lojistik regresyon skoru

```
log_reg = linear_model.LogisticRegression()
log_reg_score = cross_val_score(log_reg,X,y,cv= 10,scoring= 'accuracy').mean()
log_reg_score
```

```
0.7669856459330144
```

Bulunan lojistik regresyon skorunu sonuçlar listesine Şekil 4.14'de eklendi.

Şekil 4.14.
Lojistik regresyon skorunun eklenmesi

```
sonuçlar = []
sonuçlar.append(log_reg_score)
sonuçlar
```

```
[0.7669856459330144]
```

Doğrusal destek vektör regresyon skoru Şekil 4.15'de bulundu.

Şekil 4.15.
Doğrusal destek vektör regresyon skoru

```
linear_svm = svm.SVC(kernel = 'linear')
linear_svm_score = cross_val_score(linear_svm,X,y,cv = 10,scoring = 'accuracy').mean()
linear_svm_score
```

```
0.7656527682843473
```

Şekil 4.16'da görüldüğü üzere sonuçlar listesine eklendi.

Şekil 4.16.
Doğrusal destek vektör regresyon skorunun listeye eklenmesi

```
sonuçlar.append(linear_svm_score)
sonuçlar
```

```
[0.7669856459330144, 0.7656527682843473]
```

Lojistik regresyon algoritması ile bulunan skor doğrusal vektör algoritması ile bulunan skordan daha başarılı çıkmıştır. Bu iki algoritma arasından lojistik regresyon tercih edilecektir. En iyi sonucu veren model Şekil 4.17'de kaydedildi.

Şekil 4.17.
Modelin kaydedilmesi

```
import pickle
filename = 'diabets.sav'
log_reg.fit(X,y)
pickle.dump(log_reg,open(filename,'wb'))
```

Kaydedilen model Şekil 4.18'de çağrıldı.

Şekil 4.18
Modelin çağrılması

```
loaded_model = pickle.load(open(filename,'rb'))
loaded_model
```

```
LogisticRegression()
```

Çağrılan modeli lojistik regresyon modeli ile tahmin için Şekil 4.19'da kullanıldı. Glikoz değeri 70, vücut kitle indeksi 60 ve yaşı 50 olan bir hasta için lojistik regresyon tahmini yapıldı. İki çıktı değerimiz bulunmaktadır. Bunlardan 1 değeri diyabet hastası olduğunu, 0 değeri ise diyabet hastası olmadığını gösterir. Lojistik regresyon modeline göre bu hastanın diyabet hastası olduğu tahmin edildi.

Şekil 4.19
Modelin kullanılması

```
Glucose = 70
BMI = 60
Age = 50
prediction = loaded_model.predict([[Glucose, BMI, Age]])
```

```
prediction
```

```
array([1], dtype=int64)
```

Lojistik regresyon modeli ile başka bir tahmin Şekil 4.20'de yapıldı. Glikoz değeri 40, vücut kitle indeksi 40 ve yaşı 40 olan bir hasta için lojistik regresyon modeline göre bu hastanın diyabet hastası olmadığı tahmin edildi.

Şekil 4.20
Modelin başka bir tahminde kullanılması

```
Glucose = 40
BMI = 40
Age = 40
prediction = loaded_model.predict([[Glucose, BMI, Age]])
```

```
prediction
```

```
array([0], dtype=int64)
```