T.C

DOKUZ EYLÜL ÜNİVERSİTESİ
FEN FAKÜLTESİ

PROJE SON RAPOR

2020280001 – A. İclal AKDUMAN

2019280066 – Kamuran ÖZMEN

İÇİNDEKİLER

GİRİŞ **4**
PROJENİN AMACI **4**
VERİ HAZIRLAMA **4**
Veri Setinin Tanıtımı 4
Keşifçi Veri Analizi ve Veri Ön İşleme 5
Veri Görselleştirme 6
MODEL SEÇİMİ VE VERİ SETİNE UYGULANMASI **8**
Model Seçimi 8
Modellerin Uygulanması 8
Lojistik Regresyon 8
MLP (Multi-Layer Perceptron) 10
DNN (Deep Neural Networks) 11
SVM (Support Vector Machine) 13
XGBOOST (Extreme Gradient Boosting) 14
MODEL DEĞERLENDİRME **15**
Model Karşılaştırılması 15
WEB ARAYÜZÜ **16**
SONUÇ **17**
KAYNAKÇA **18**

GÖRSELLER LİSTESİ

1: Veri seti özellikleri....5
2: Veri setine istatistiksel bakış....5
3: Sütunlardaki benzersiz değişkenler....6
4: Verilerin dağılımı....6
5: Yaşa bağlı inme verilerinde aykırı değer grafiği....7
6: Vücut kitle endeksine bağlı inme verilerinde aykırı değer grafiği....7
7: Korelasyon matrisi....8
8: Aykırı değer silme....9
9: Lojistik regresyon model oluşturma....9
10: Lojistik regresron doğruluk oranı....10
11. MLP model....11
12: MLP doğruluk oranı....11
13: DNN model....12
14: DNN model eğitimi....12
15: DNN epoch sonuçları....12
16: DNN doğruluk oranı....13
17: SVM model....14
18: SVM doğruluk oranı....14
19: XGBOOST model....15
20: XGBOOST doğruluk oranı....15
21: Web arayüzü....16
22: Web arayüzü....16
23: Bilinen veri ile tahmin denemesi....17

BEYİN FELCİ RİSK TESPİTİ

BİL4112-YAPAY ZEKA

GİRİŞ

Bu rapor proje kapsamında gerçekleştirmiş olduğumuz adımları kapsamlı bir şekilde anlatmaktadır. Bu süreç veri setini tanımlama, keşifçi veri analizi ve ön işleme, model seçimi ve eğitim, model değerlendirmesi ve son olarak web arayüzüne entegrasyon aşamalarını içermektedir.

PROJENİN AMACI

Veri tabanındaki verileri analiz ederek, beyin felci (inme) hastalığını önceden tespit etmek mümkündür. Öncelikle hipertansiyon, kalp hastalığı, yaş, ortalama glikoz seviyesi ve BMI gibi risk faktörlerini dikkate alarak bir risk profili oluşturulabilir. Bu risk profiline göre yüksek risk altındaki bireyleri belirleyerek daha detaylı tetkikler yapılması sağlanabilir.

VERİ HAZIRLAMA

Veri Setinin Tanıtımı

Toplamda 4982 satırdan oluşan veri setinde 8'i kategorik 3'ü numerik olmak üzere 11 adet değişken mevcuttur. Bu değişkenler sırasıyla;

1. Cinsiyet: Hastanın cinsiyeti, "Kadın" veya "Erkek".
2. Yaş: Hastanın yaşı.
3. Hipertansiyon: Hastanın hipertansiyonu olup olmadığını belirten değer, yoksa 0 varsa1.
4. Kalp Hastalığı: Hastanın kalp hastalığı olup olmadığını belirten değer, yoksa 0 varsa1.
5. Evlilik Durumu: Hastanın medeni durumunu belirten değer, "Evet" veya "Hayır".
6. İş Türü: Hastanın iş durumunu belirtir. "Çocuk", "Devlet dairesi", "Hiç çalışmamış", "Özel" veya "Serbest meslek".
7. İkamet Türü: Hastanın ikamet ettiği yerin türünü belirten değer. "Kentsel" veya "Kırsal".
8. Ortalama Glikoz Seviyesi: Hastanın kanında bulunan ortalama glikoz seviyesi.
9. BMI: Hastanın vücut kitle endeksi.
10. Sigara İçme Durumu: Hastanın sigara tüketimi belirten değer, "Önceden içmiş", "Hiç içmemiş" veya "Sigara içiyor".
11. Beyin Felci (İnme): Hastanın inme geçirip geçirmediğini belirten değer, geçirmemişse 0 geçirmişse 1.

şeklindedir.

Keşifçi Veri Analizi ve Veri Ön İşleme

Veri setindeki gözlem ve değişken sayısı belirlenmiş olup bu değişkenlerin veri tipleri incelenmiştir. Veri setine ait özellikler ve hedef değişkenler belirlenmiştir. Eksik veriler ve kontrol edilmiştir ve eri setinde eksik değer bulunmadığı için herhangi bir işlem uygulanmamıştır. Veri setinin istatistiksel özelliklerine göz atılmış ve temel istatistiksel özellikleri kaydedilmiştir. Veri setinde bulunan sütunlar ve bu sütunlardaki benzersiz değerlere bakılmıştır.

```
#Gözlem sayısı ve değişken sayısı için:
df.shape
```

```
(4981, 11)
```

```
#Değişkenlerin veri tipleri ve eksik gözlem kontrolü için:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   gender          4981 non-null   object 
 1   age             4981 non-null   float64
 2   hypertension    4981 non-null   int64  
 3   heart_disease   4981 non-null   int64  
 4   ever_married    4981 non-null   object 
 5   work_type       4981 non-null   object 
 6   Residence_type  4981 non-null   object 
 7   avg_glucose_level  4981 non-null   float64
 8   bmi             4981 non-null   float64
 9   smoking_status  4981 non-null   object 
 10  stroke          4981 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 428.2+ KB
```

1: Veri seti özellikleri

```
#İstatistiksel ölçüm
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	4981.0	43.419859	22.662755	0.08	25.00	45.00	61.00	82.00
hypertension	4981.0	0.096165	0.294848	0.00	0.00	0.00	0.00	1.00
heart_disease	4981.0	0.055210	0.228412	0.00	0.00	0.00	0.00	1.00
avg_glucose_level	4981.0	105.943562	45.075373	55.12	77.23	91.85	113.86	271.74
bmi	4981.0	28.498173	6.790464	14.00	23.70	28.10	32.60	48.90
stroke	4981.0	0.049789	0.217531	0.00	0.00	0.00	0.00	1.00

2: Veri setine istatistiksel bakış

```
#Yukarıdaki gibi tek tek göstermek yerine beraber gösterimi
df_uniq=df[['gender', 'hypertension', 'heart_disease', 'ever_married','work_type', 'Residence_type',
            'smoking_status', 'stroke']]

for i in df_uniq.columns:
  print(df_uniq[i].unique())
```

```
['Male' 'Female']
[0 1]
[1 0]
['Yes' 'No']
['Private' 'Self-employed' 'Govt_job' 'children']
['Urban' 'Rural']
['formerly smoked' 'never smoked' 'smokes' 'Unknown']
[1 0]
```

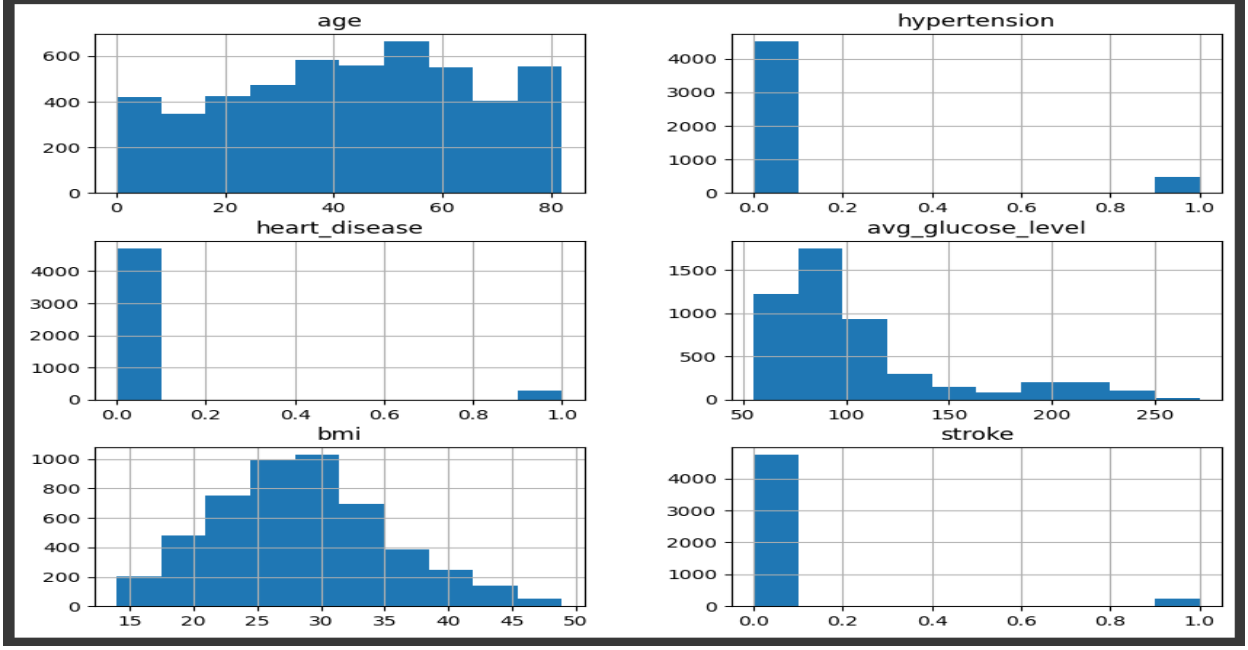
3: Sütunlardaki benzersiz değişkenler

Veri Görselleştirme

Veri setinin görselleştirilmesi, veri setinin dağılımı ve değişkenler arası ilişkilerin değerlendirilmesi için büyük önem taşımaktadır. Bu sebeple veri görselleştirme adımı da gerçekleştirilmiş olup ilişkiler incelenmiştir.

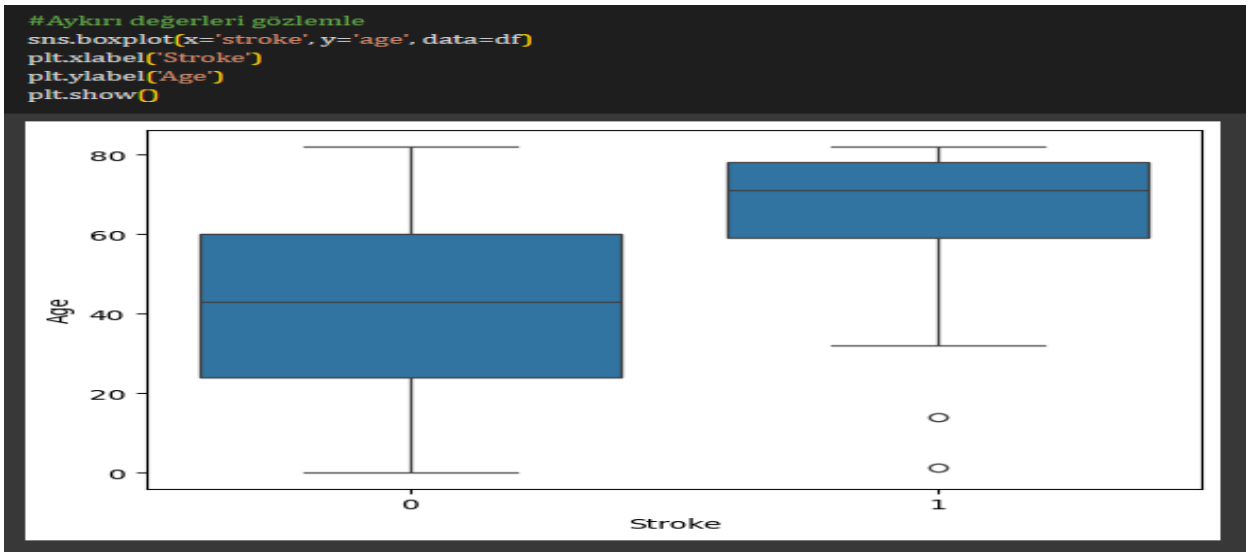
Şekil 4'te veri setindeki sayısal değişkenlerin dağılımı verilmiştir. Bu dağılımlar incelendiğinde; yaş dağılımının 40 yaş ve üstü grupta yoğunluk gösterdiği, vücut kitle endeksinin (BMI) 25 ile 30 değerleri arsında yoğunlukta olduğu, veri setinde kalp krizi geçiren, hipertansiyonu olan ve inme geçiren kişilerin geçirmeyen kişilere göre sayıca çok daha az olduğu gözlemlenmiştir.

```
#Sayısal değişkenlerin dağılımı
df.hist(bins=10, figsize=(10,8))
plt.show()
```



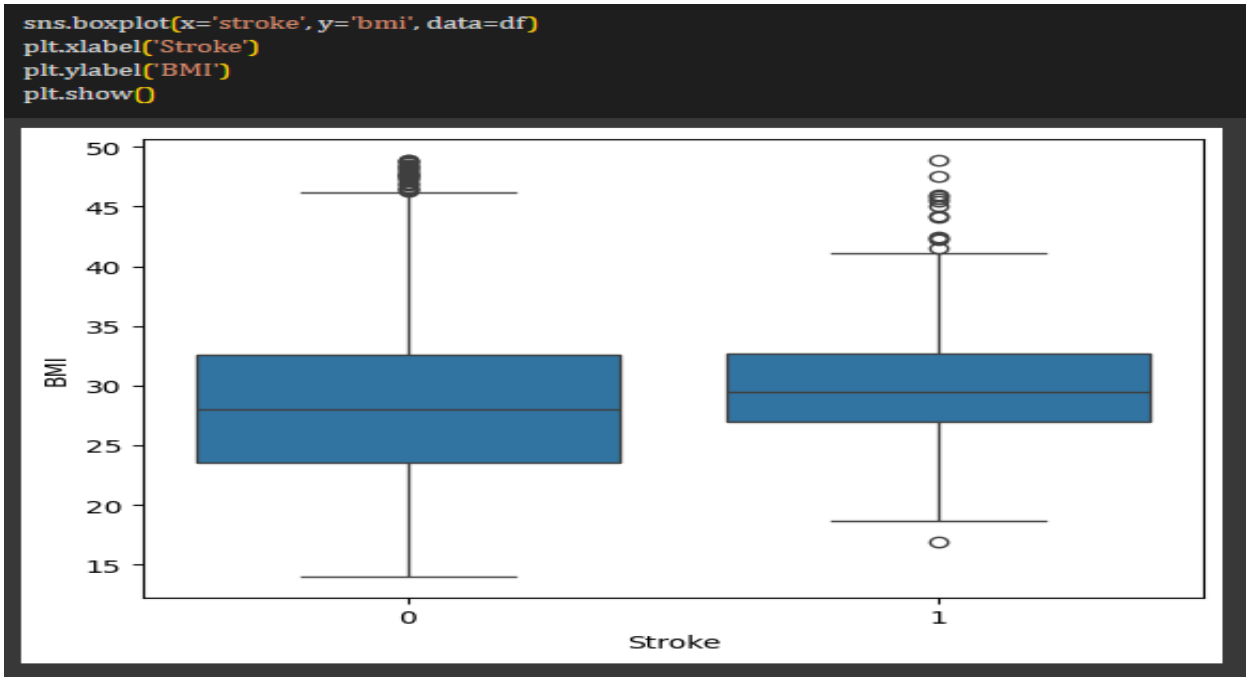
4: Verilerin dağılımı

Şekil 5’te kutu grafiği kullanılarak yaşa bağlı olarak inme geçirme durumu incelenmiştir. Grafik incelendiğinde inme geçiren kişilerin yaş dağılımı 60-80 yaş grubu arasında değişmektedir. Fakat grafikte gözlemlendiği üzere daha genç yaş grubu olan 0-20 yaş grubunda da inme geçiren kişiler gözlenmiştir. İnme geçirmeyen kişilerin yaş dağılımı ise 20 ila 60 yaşları arasında dağılım göstermektedir.



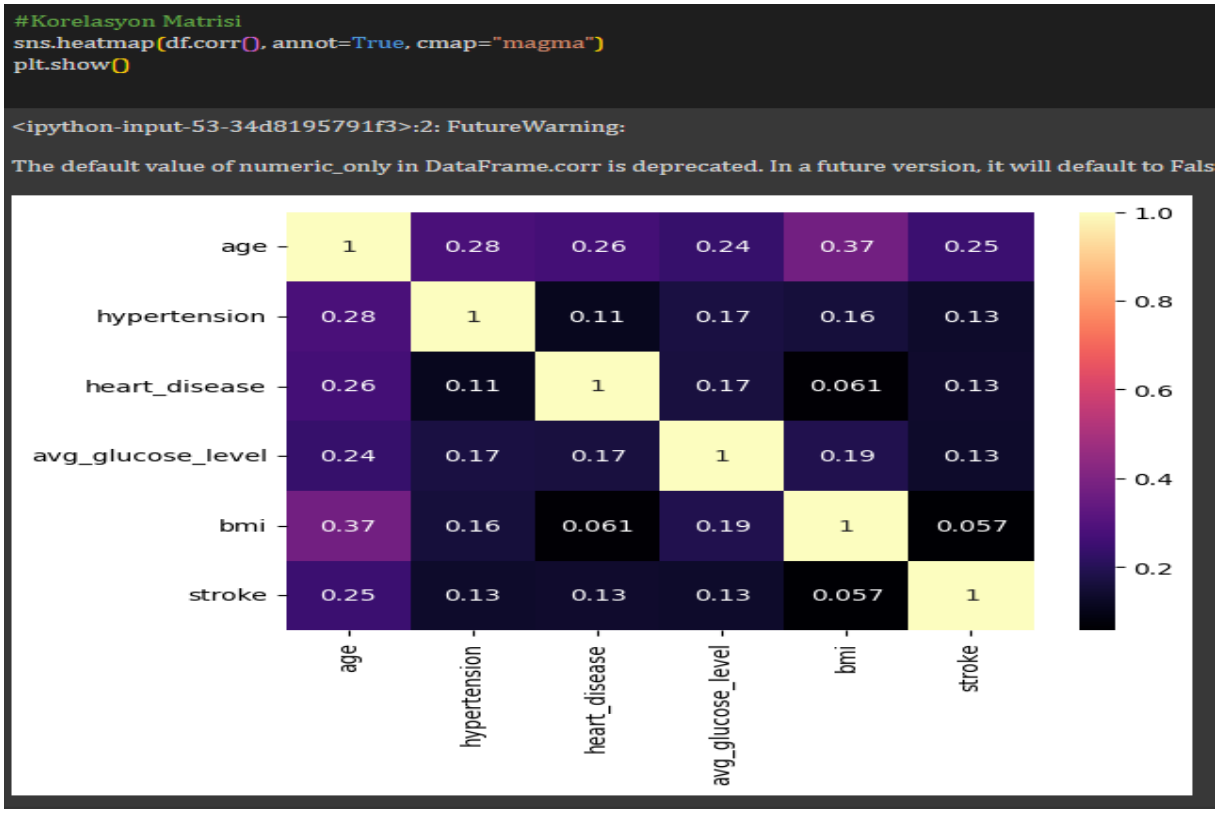
5: Yaşa bağlı inme verilerinde aykırı değer grafiği

Şekil 6'da, BMI'ye göre inme geçirme durumu kutu grafiği ile incelenmiştir. İnme geçirmeyenler çoğunlukla 20-35 yaş aralığında, 45 yaş üstünde ise aykırı değerler görülmektedir. İnme geçirenler ise genelde 25-35 yaş aralığında, 20 yaş altı ve 40 yaş üstünde aykırı değerler bulunmaktadır. Bu, inme riskinin yaşla ilişkili olduğunu göstermektedir.



6: Vücut kitle endeksine bağlı inme verilerinde aykırı değer grafiği

Şekil 7'deki korelasyon matrisi incelendiğinde, en güçlü ilişkinin yaş ve BMI arasında 0.37 ile olduğu görülmüştür. Yaş ile hipertansiyon arasında 0.28, en düşük ilişki ise kalp krizi geçirme durumu ile BMI arasında 0.061 olarak tespit edilmiştir. Bu bulgular, yaş ile BMI arasında orta derecede pozitif, yaş ile hipertansiyon arasında zayıf bir ilişki olduğunu, kalp krizi geçirme durumu ile BMI arasında ise neredeyse hiç ilişki olmadığını göstermektedir.



7: Korelasyon matrisi

MODEL SEÇİMİ VE VERİ SETİNE UYGULANMASI

Model Seçimi

Veri setimize uygulanabilecek beş model belirlenmiştir. Bu modeller:

- Lojistik Regresyon
- MLP (Multi-Layer Perceptron)
- DNN (Deep Neural Networks)
- SVM (Support Vector Machine)
- XGBOOST (Extreme Gradient Boosting)

Modellerin Uygulanması

Lojistik Regresyon

Lojistik regresyon, ikili sınıflandırma problemlerinde yaygın bir biçimde tercih edilen temel bir makine öğrenimi modelidir. Modelin uygulanması öncesinde, veri setinde keşifçi

veri analizinde tespit edilen aykırı değerlerin, modelin performansını olumsuz etkileyebileceği düşünülerek bu değerler veri setinden çıkarılmıştır. Daha sonra, veri setinde bulunan kategorik değişkenler, modelin işleyebileceği bir formata dönüştürülebilmesi için label encoding tekniği kullanılmıştır.

Modelin uygulanabilmesi için, veriler özelliklere (bağımsız değişkenlere) ve hedef değişkene (bağımlı değişkene) ayrılmıştır. Bu ayrıştırma işleminden sonra, veri seti %80 eğitim ve %20 test kümesi olacak şekilde iki farklı kümeye ayrılmıştır. Ardından, model oluşturulmuş, eğitilmiş ve test edilmiştir. Sonuçlar incelendiğinde, model üzerinden yapılan tahminlerin doğruluğunun %95,86 olduğu gözlemlenmiştir.

Aykırı değerleri silme

```
Q1 = df['bmi'].quantile(0.25)
Q3 = df['bmi'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier_indices = (df['bmi'] < lower_bound) | (df['bmi'] > upper_bound)
df = df[~outlier_indices]

Q1 = df['avg_glucose_level'].quantile(0.25)
Q3 = df['avg_glucose_level'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier_indices = (df['avg_glucose_level'] < lower_bound) | (df['avg_glucose_level'] > upper_bound)
df = df[~outlier_indices]

Q1 = df['age'].quantile(0.25)
Q3 = df['age'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outlier_indices = (df['age'] < lower_bound) | (df['age'] > upper_bound)
df = df[~outlier_indices]
```

8: Aykırı değer silme

Kategorik değişkenleri dönüştürme

```
label_encoder = LabelEncoder()
df['gender'] = label_encoder.fit_transform(df['gender'])
df['ever_married'] = label_encoder.fit_transform(df['ever_married'])
df['work_type'] = label_encoder.fit_transform(df['work_type'])
df['Residence_type'] = label_encoder.fit_transform(df['Residence_type'])
df['smoking_status'] = label_encoder.fit_transform(df['smoking_status'])
```

Verileri özelliklere ve hedefe bölme

```
X = df.drop("stroke", axis=1)
y = df["stroke"]
```

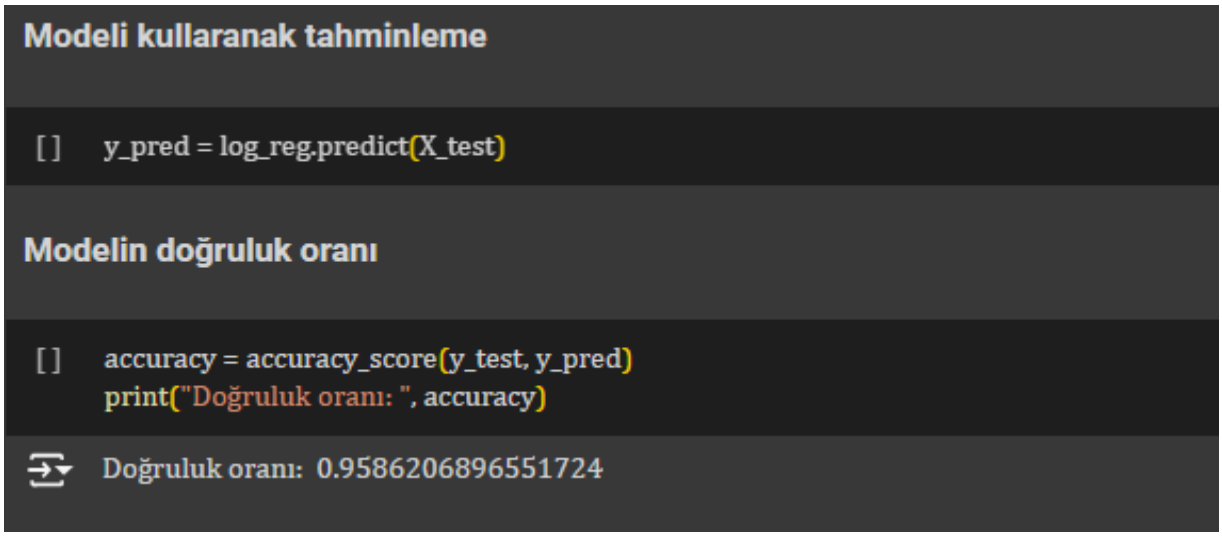
Verileri test ve eğitim kümesi olarak ayırma

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Modeli oluşturma ve eğitme

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

9: Lojistik regresyon model oluşturma



10: Lojistik regresron doğruluk oranı

MLP (Multi-Layer Perceptron)

MLP, yani Çok Katmanlı Algılayıcı (Multi-Layer Perceptron), yapay sinir ağlarının en temel ve yaygın formlarından biridir. MLP, birbirine bağlı nöronlardan oluşan birden fazla katmana sahiptir.

Bu modelin oluşturulmasında, kategorik değişkenlerin dönüşümü için one hot encoding tekniği tercih edilmiştir. Model için özelliklerin ve hedef değişkeninin belirlenmesinin ardından, veri seti %70 eğitim ve %30 test verisi olacak şekilde iki farklı kümeye ayrılmıştır. Veri ölçeklendirme işlemi için RobustScaler yöntemi seçilmiştir. Bu seçimdeki ana neden, RobustScaler yönteminin diğer ölçeklendirme yöntemlerine göre aykırı değerlere karşı daha dirençli olmasıdır. Bu yöntem, aykırı değerlerin etkisini azaltırken veri setinin genel dağılımını koruyarak veri dengesini sağlamada etkili bir yöntem olarak kabul edilir. Dolayısıyla, RobustScaler yöntemi aykırı değerlere karşı daha dirençli bir ölçeklendirme yöntemi olarak tercih edilmiştir.

Model, derin öğrenme kütüphanesi olan Keras kullanılarak oluşturulmuştur. Modelde kullanılan parametreler şu şekildedir:

- hidden_layer_sizes: Gizli katmanların sayısını ve her katmandaki nöron sayısını belirtir. Toplamda dört gizli katman belirlenmiştir. Her bir katmanda sıraysıyla 32,16,4 ve 2 nöron bulunmaktadır.
- max_iter: Modelin eğitimi sırasında maksimum iterasyon sayısını belirler.
- activation: Gizli katmanlarda kullanılacak olan aktivasyon fonksiyonunu berliler. Modelde en yaygın kullanılan fonksiyonlardan biri olan ReLU kullanılmıştır.
- solver: Bu parametre optimize edici algortimayı belirler. Modelde etkili bir optimize edici olan adam kullanılmıştır.

Sonrasında model eğitilip test edilmiştir ve modelin tahmin doğruluğu %92,97 olarak gözlemlenmiştir.

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler

df = pd.get_dummies(df, columns=['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status'])

X = df.drop("stroke", axis=1)
y = df["stroke"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

robust = RobustScaler()
X_train_scaled = robust.fit_transform(X_train)
X_test_scaled = robust.transform(X_test)
```

11. MLP model

```
mlp = MLPClassifier(hidden_layer_sizes=(32,16,4,2),  max_iter=1000, activation='relu', solver='adam', random_state=42)

mlp.fit(X_train_scaled, y_train)
```

MLPClassifier

MLPClassifier(hidden_layer_sizes=(32, 16, 4, 2), max_iter=1000, random_state=42)

```
y_pred = mlp.predict(X_test_scaled)

acc = accuracy_score(y_test, y_pred)
print('acc: ', {acc})
```

```
acc:  {0.9297658862876255}
```

12: MLP doğruluk oranı

DNN (Deep Neural Networks)

Yapay sinir ağları, insan beynini örnek alarak öğrenme sürecini matematiksel olarak modellemek amacıyla geliştirilmiştir. Bu çalışmada, modelin uygulanmasında çeşitli ön işleme adımları gerçekleştirilmiştir. İlk olarak, kategorik değişkenler, label encoding tekniği kullanılarak numerik değişkenlere dönüştürülmüştür. Ardından, veri seti, özellikler (bağımsız değişkenler) ve hedef değişken (bağımlı değişken) olarak ayrılmıştır. Bu işlemden sonra veri setinin %75'i eğitim, %25'i test verisi olarak iki kümeye bölünmüştür. Aykırı değerlere karşı direnç sağlamak amacıyla, veriler RobustScaler yöntemi ile ölçeklendirilmiştir.

Model, Keras kütüphanesi kullanılarak oluşturulmuştur. Gizli katmanlardaki nöron sayıları ve aktivasyon fonksiyonları dikkatlice belirlenmiştir. Aşırı öğrenmeyi (overfitting) önlemek amacıyla, katmanlar arasına dropout katmanları eklenmiştir. Modelin derlenmesi aşamasında, çeşitli hiperparametreler tanımlandıktan sonra eğitim sürecine geçilmiştir. Modelin derlenmesi ve eğitimi sırasında kullanılan bazı temel parametreler şunlardır:

- Epoch: Eğitim iterasyon sayısı.
- Batch Size: Her iterasyonda işlenecek örnek sayısı.
- İyileştirme Verisi: Modelin eğitilmesini iyileştirmek için kullanılır.
- Optimizer: Modelin eğitim sırasında kullanacağı optimize edici algoritmayı belirler.

- Loss: Her eğitimden sonra elde edilen değerler ile gerçek değerler arasındaki hata farkının hesaplanmasını belirleyen parameteredir.
- Metrics: Modelin performansını değerlendirmek için kullanılacak ölçütleri belirler.

Model eğitilip tahminleme yapıldığında doğruluk oranı %94,46 olarak kaydedilmiştir.

```
label_encoder = LabelEncoder()
df['gender'] = label_encoder.fit_transform(df['gender'])
df['ever_married'] = label_encoder.fit_transform(df['ever_married'])
df['work_type'] = label_encoder.fit_transform(df['work_type'])
df['Residence_type'] = label_encoder.fit_transform(df['Residence_type'])
df['smoking_status'] = label_encoder.fit_transform(df['smoking_status'])

X = df.drop("stroke", axis=1)
y = df["stroke"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

robust = RobustScaler()
X_train_scaled = robust.fit_transform(X_train)
X_test_scaled = robust.transform(X_test)
```

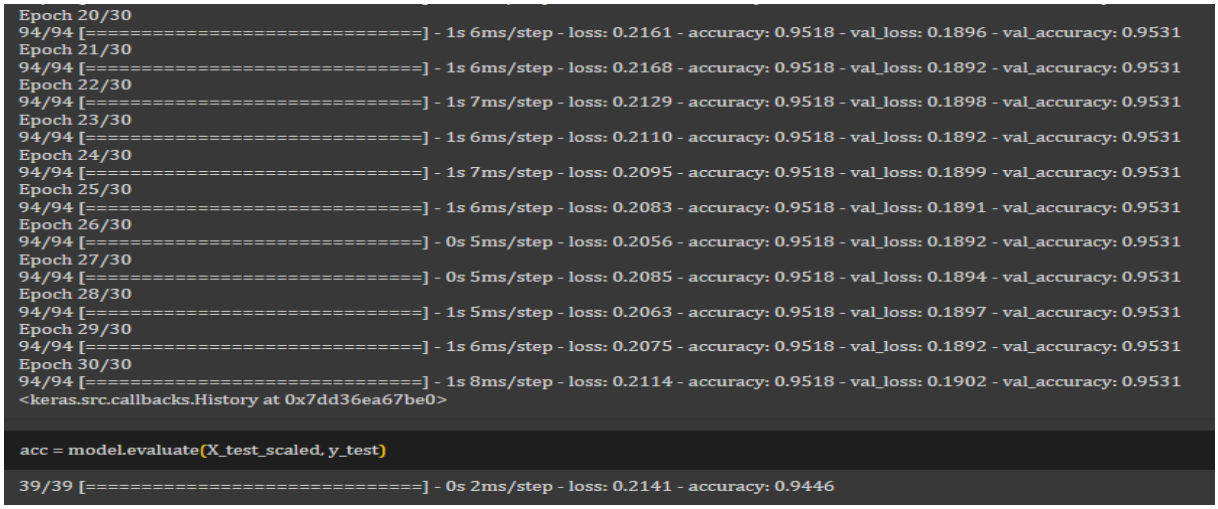
13: DNN model

```
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(X_train_scaled.shape[1], )))
model.add(Dropout(0.2))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(4, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train_scaled, y_train, epochs=30, batch_size=32, validation_split=0.2)
```

14: DNN model eğitimi

```
Epoch 1/30
94/94 [==============================] - 6s 8ms/step - loss: 0.5006 - accuracy: 0.9491 - val_loss: 0.2314 - val_accuracy: 0.9531
Epoch 2/30
94/94 [==============================] - 0s 5ms/step - loss: 0.2560 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 3/30
94/94 [==============================] - 1s 7ms/step - loss: 0.2400 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 4/30
94/94 [==============================] - 1s 7ms/step - loss: 0.2287 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 5/30
94/94 [==============================] - 1s 7ms/step - loss: 0.2273 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 6/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2241 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 7/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2146 - accuracy: 0.9518 - val_loss: 0.1896 - val_accuracy: 0.9531
Epoch 8/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2245 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 9/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2167 - accuracy: 0.9518 - val_loss: 0.1898 - val_accuracy: 0.9531
Epoch 10/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2146 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 11/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2170 - accuracy: 0.9518 - val_loss: 0.1893 - val_accuracy: 0.9531
Epoch 12/30
94/94 [==============================] - 1s 13ms/step - loss: 0.2268 - accuracy: 0.9518 - val_loss: 0.1894 - val_accuracy: 0.9531
Epoch 13/30
94/94 [==============================] - 1s 8ms/step - loss: 0.2158 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 14/30
94/94 [==============================] - 1s 12ms/step - loss: 0.2234 - accuracy: 0.9518 - val_loss: 0.1893 - val_accuracy: 0.9531
Epoch 15/30
94/94 [==============================] - 1s 9ms/step - loss: 0.2167 - accuracy: 0.9518 - val_loss: 0.1892 - val_accuracy: 0.9531
Epoch 16/30
94/94 [==============================] - 1s 9ms/step - loss: 0.2197 - accuracy: 0.9518 - val_loss: 0.1900 - val_accuracy: 0.9531
Epoch 17/30
94/94 [==============================] - 1s 8ms/step - loss: 0.2141 - accuracy: 0.9518 - val_loss: 0.1891 - val_accuracy: 0.9531
Epoch 18/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2116 - accuracy: 0.9518 - val_loss: 0.1894 - val_accuracy: 0.9531
Epoch 19/30
94/94 [==============================] - 1s 6ms/step - loss: 0.2186 - accuracy: 0.9518 - val_loss: 0.1891 - val_accuracy: 0.9531
Epoch 20/30
```

15: DNN epoch sonuçları



16: DNN doğruluk oranı

SVM (Support Vector Machine)

Destek Vektör Makineleri (Support Vector Machines, SVM), genellikle sınıflandırma problemlerinde kullanılan gözetimli öğrenme yöntemlerinden biridir. İlk olarak, veri setinde yer alan kategorik değişkenler, label encoding tekniği ile numerik değerlere dönüştürülmüştür.

Aykırı değerlerin model performansını olumsuz etkilemesini önlemek amacıyla, bu değerler minimum ve maksimum sınırlara çekilmiştir. Bu işlem için 1. ve 3. çeyrekler (Q1 ve Q3) arasındaki fark olan IQR (Interquartile Range) kullanılmıştır. IQR değeri, 1.5 ile çarpılarak aykırı değerlerin sınırları belirlenmiştir. Q1 - 1.5IQR formülü ile minimum sınır, Q3 + 1.5IQR formülü ile maksimum sınır belirlenmiştir. Bu çarpan, dağılımın genel yapısını bozmamak için 1.5 olarak seçilmiştir. Aykırı değerler bu sınırlar içerisine dahil edildikten sonra, veriler özellikler (bağımsız değişkenler) ve hedef değişken (bağımlı değişken) olarak ayrılmıştır. Veri seti, %70 eğitim ve %30 test kümesi olmak üzere iki ayrı küme oluşturulmuştur. Veri ölçeklendirme işlemi, StandardScaler yöntemi kullanılarak gerçekleştirilmiştir. StandardScaler, verilerin ortalama etrafında birim varyans ile ölçeklendirilmesini sağlar ve bu da modelin daha iyi performans göstermesine yardımcı olur.

Modelin oluşturulmasında, verilerin özellik uzayında nasıl dönüştürüleceğini ve sınıflandırılacağını belirleyen kernel parametresi, doğrusal (linear) olarak belirlenmiştir. Bu, modelin doğrusal bir karar sınırı kullanarak sınıflandırma yapmasını sağlar. Model, scikit-learn kütüphanesindeki SVC sınıfı kullanılarak oluşturulmuştur. Model oluşturulduktan sonra, eğitim verisi üzerinde eğitilmiş ve ardından test verisi üzerinde tahminleme yapılmıştır. Eğitim sonucunda modelin doğruluk oranı %94.78 olarak gözlemlenmiştir.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import  SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

label_encoder = LabelEncoder()
df['gender'] = label_encoder.fit_transform(df['gender'])
df['ever_married'] = label_encoder.fit_transform(df['ever_married'])
df['work_type'] = label_encoder.fit_transform(df['work_type'])
df['Residence_type'] = label_encoder.fit_transform(df['Residence_type'])
df['smoking_status'] = label_encoder.fit_transform(df['smoking_status'])

outliner = ['age','bmi','avg_glucose_level']
for i in outliner:
   Q1,Q3=np.quantile(df[i],q=[0.25,0.75])
   IQR=Q3-Q1
   lower_bound = Q1 - (1.5 * IQR)
   upper_bound = Q3 + (1.5 * IQR)
   df[i]=np.clip(df[i],lower_bound,upper_bound)

X = df.drop("stroke", axis=1)
y = df["stroke"]
```

17: SVM model

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

svm = SVC(kernel='linear')
svm.fit(X_train_scaled, y_train)
```

```
SVC
SVC(kernel='linear')
```

```
svm_pred = svm.predict( X_test_scaled)

accuracy = accuracy_score(y_test, svm_pred)
print("Doğruluk oranı: ", accuracy)
```

```
Doğruluk oranı:  0.9478260869565217
```

18: SVM doğruluk oranı

XGBOOST (Extreme Gradient Boosting)

Karar ağacı temelli Gradient Boosting algoritmasının çeşitli düzenlemeler ile optimize edilmiş yüksek performanslı halidir.

Bu modelde kategorik değişkenler one hot encoding tekniği ile dönüştürülmüştür. Sonrasında ise veriler özellikler ve hedef değişken olmak üzere ayrılmıştır. Veri seti %80 eğitim %20 test olacak şekilde iki kümeye ayrıldıktan sonra StandardScaler yöntemi ile özelleştirilmiştir. Model oluşturma aşamasında ise ikili sınıflandırma problemlerinde kullanılan logloss metriği kullanılmıştır. Model eğitildikten sonra tahminlemesi gerçekleştirilmiştir. Bu tahminlemenin doğruluk oranı %94,28 olarak gözlemlenmiştir.

```
df = pd.get_dummies(df, columns=['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status'])

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

X = df.drop('stroke', axis=1)
y = df['stroke']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

19: XGBOOST model

```
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

xgb_model.fit(X_train_scaled, y_train)

▸ XGBClassifier

y_pred = xgb_model.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

Accuracy: 0.9428284854563691
```

20: XGBOOST doğruluk oranı

MODEL DEĞERLENDİRME

Model Karşılaştırılması

MODEL	DOĞRULUK ORANI
LOJİSTİK REGRESYON	%95,86
MLP	%92,97
DNN	%94,46
SVM	%94,78
XGBOOST	%94,28

Veri setimize uyguladığımız 5 adet modelden en iyi sonucu veren model lojistik regresyon modeli olmuştur.

WEB ARAYÜZÜ

Eğitilmiş olan lojistik regresyon modeli web arayüzüne entegre edilmek için joblib kütüphanesi aracılığıyla kaydedilmiştir. Entegrasyon işlemi için bir Python dosyası oluşturulur ve bu dosyaya kaydedilen model yüklenmiştir. Daha sonrasında web arayüzünün oluşması için streamlit kütüphanesi kullanılarak başlık, kullanıcıdan veri girişi ve tahminleme işlemi gibi işlemler gerçekleştirilmiştir. Uygulama streamlit run [dosyaadı].py şeklinde çalıştırıldığında başarılı bir sonuç elde edilmiştir.

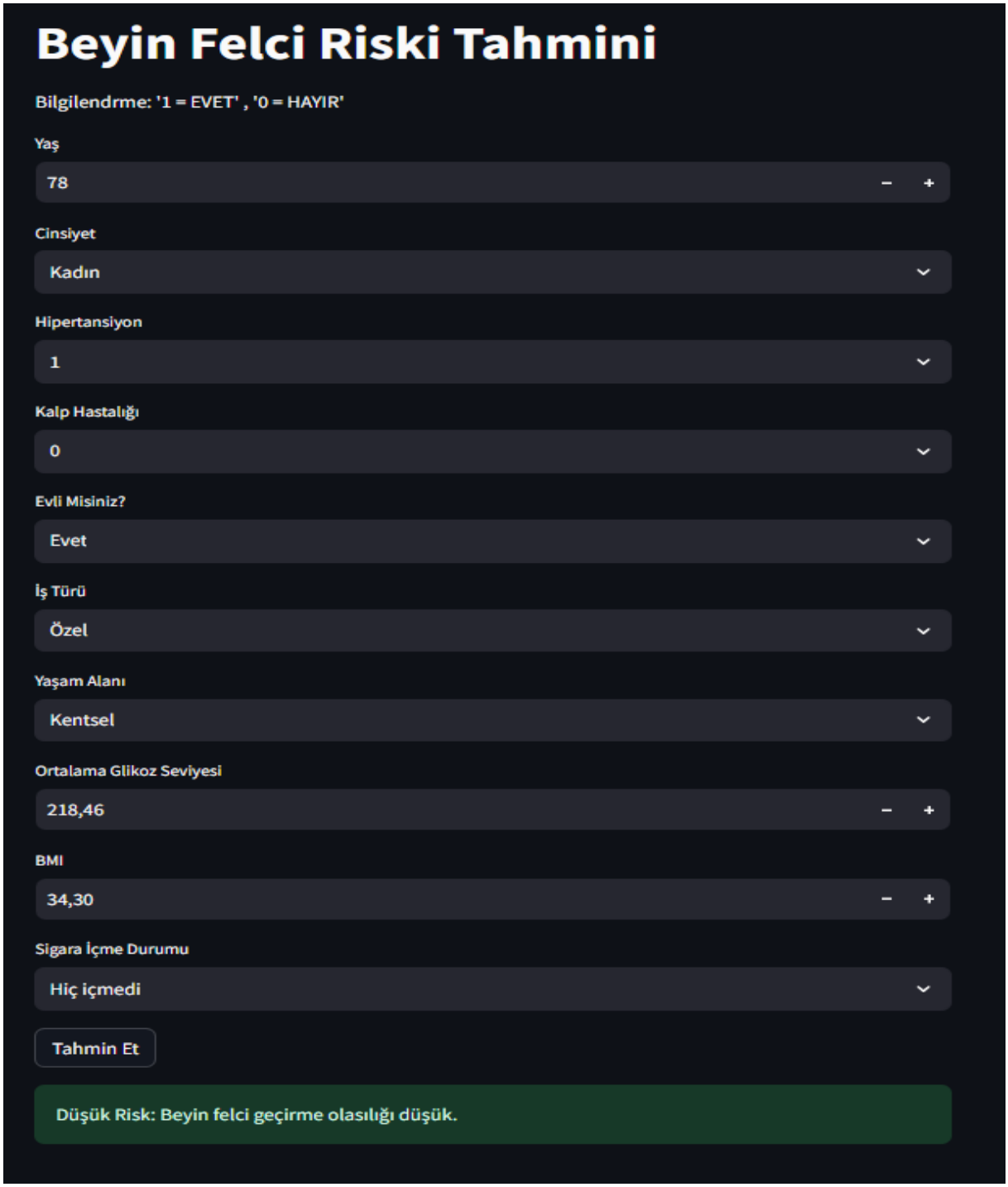
app.py

```python
import numpy as np
import pandas as pd
import streamlit as st
import joblib

loaded_model = joblib.load('logistic_regression_model.pkl')


st.title('Beyin Felci Riski Tahmini')

st.write("Bilgilendrme: '1 = EVET' , '0 = HAYIR'")

age = st.number_input('Yaş', min_value=0, max_value=150, value=25)
gender = st.selectbox('Cinsiyet', ['Erkek', 'Kadın'])
hypertension = st.selectbox('Hipertansiyon', [0, 1])
heart_disease = st.selectbox('Kalp Hastalığı', [0, 1])
ever_married = st.selectbox('Evli Misiniz?', ['Evet', 'Hayır'])
work_type = st.selectbox('İş Türü', ['Çalışmıyor', 'Özel', 'Serbest', 'Devlet', 'Çocuk'])
residence_type = st.selectbox('Yaşam Alanı', ['Kırsal', 'Kentsel'])
avg_glucose_level = st.number_input('Ortalama Glikoz Seviyesi', min_value=0.0, max_value=400.0, value=100.0)
bmi = st.number_input('BMI', min_value=0.0, max_value=100.0, value=25.0)
smoking_status = st.selectbox('Sigara İçme Durumu', ['Hiç içmedi', 'Eskiden içti', 'Şu anda içiyor'])

```

21: Web arayüzü

```python
gender_mapping = {'Erkek': 1, 'Kadın': 0}
ever_married_mapping = {'Evet': 1, 'Hayır': 0}
work_type_mapping = {'Çalışmıyor': 0, 'Özel': 1, 'Serbest': 2, 'Devlet': 3, 'Çocuk': 4}
residence_type_mapping = {'Kırsal': 0, 'Kentsel': 1}
smoking_status_mapping = {'Hiç içmedi': 0, 'Eskiden içti': 1, 'Şu anda içiyor': 2}

input_data = np.array([[
    age,
    gender_mapping[gender],
    hypertension,
    heart_disease,
    ever_married_mapping[ever_married],
    work_type_mapping[work_type],
    residence_type_mapping[residence_type],
    avg_glucose_level,
    bmi,
    smoking_status_mapping[smoking_status]
]], dtype=float)

if st.button('Tahmin Et'):
    prediction = loaded_model.predict(input_data)
    if prediction[0] == 1:
        st.error('Yüksek Risk: Beyin felci geçirme olasılığı yüksek.')
    else:
        st.success('Düşük Risk: Beyin felci geçirme olasılığı düşük.')
```

22: Web arayüzü



23: Bilinen veri ile tahmin denemesi

SONUÇ

Bu raporda veri setine uygulanan 5 farklı model değerlendirilmiştir ve en iyi sonucu veren model web arayüzüne entegre edilmiştir. Bu süreçte sırasıyla veri tanıtımı, keşifçi veri analizi ve veri ön işleme, model seçimi ve eğitilmesi, model değerlendirme ve web arayüzüne entegrasyon işlemleri detaylı bir şekilde ele alınmıştır. Sonuç olarak kullanıcıların erişimine sunulan bir tahmin sistemi oluşturulmuştur. Tahmin sistemi bilinen veri ile denendiğinde doğru sonuç verdiği gözlemlenmiştir.

KAYNAKÇA

Akça, M. F. (2020, Ağustos). *Nedir Bu Destek Vektör Makineleri? (Makine Öğrenmesi Serisi-2)*. Medium: https://medium.com/deep-learning-turkiye/nedir-bu-destek-vekt%C3%B6r-makineleri-makine-%C3%B6%C4%9Frenmesi-serisi-2-94e576e4223e adresinden alındı

BEŞTAŞ, M. (tarih yok). *Keşifçi Veri Analizi ile Eczane Satış Analizi ve Satış Tahmini*. Makale Sistemi: http://www.makalesistemi.com/panel/files/manuscript_files_publish/e61942b4897972dd6a60f8037db34c7c/9b0fbb123bdabcfa3e8f0755b1556915/2ee40a45e3b45f4.pdf adresinden alındı

Boz, U. (2020, Kasım). *OpenCV ile Derin Öğrenme — Temel Bilgiler*. Medium: https://umut-boz.medium.com/opencv-ile-derin-%C3%B6%C4%9Frenme-temel-bilgiler-ac93036f6690 adresinden alındı

Duru, M. C. (2024, Mart). *Özellik Ölçeklendirme: StandardScaler, RobustScaler ve MinMaxScaler Karşılaştırması*. Medium: Özellik Ölçeklendirme: StandardScaler, RobustScaler ve MinMaxScaler Karşılaştırması adresinden alındı

Güler, E. (2023, Haziran). *XGBoost Algoritması*. Medium: https://efecanxrd.medium.com/xgboost-algoritmas%C4%B1-6703b14efd5 adresinden alındı

J, N. H. (2010, December 18). *Gender differences in coronary heart disease*. National Library of Medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3018605/ adresinden alındı

Lojistik Regresyon. (tarih yok). aws.amazon: https://aws.amazon.com/tr/what-is/logistic-regression/#:~:text=Lojistik%20regresyon%2C%20iki%20veri%20fakt%C3%B6r%C3%BC,gibi%20s%C4%B1n%C4%B1rl%C4%B1%20say%C4%B1da%20sonucu%20vard%C4%B1r. adresinden alındı

Sezer, Ş. (2021, Haziran). *Streamlit ile Web Uygulamalarına Giriş*. Medium: https://medium.com/machine-learning-t%C3%BCrkiye/streamlit-ile-web-uygulamalar%C4%B1na-giri%C5%9F-7093e749973f adresinden alındı

Şirin, E. (2017). *Büyük Veri Ön-İşleme (Makale Notları)*. Veri Bilimi Okulu: https://www.veribilimiokulu.com/buyuk-veri-on-isleme-makale-notlari/#:~:text=Veri%20%C3%96n%20%C4%B0%C5%9Fleme%20(Data%20preprocessing,%2C%20boyut%20indirgeme%20vb.%20i%C5%9Flemlerdir. adresinden alındı

Tüzemen, B. (2023, Aralık). *MLP (Çok Katmanlı Algılayıcı) Yapay Sinir Ağları*. Medium: https://medium.com/@baristuzemenn/mlp-%C3%A7ok-katmanl%C4%B1-alg%C4%B1lay%C4%B1c%C4%B1-yapay-sinir-a%C4%9Flar%C4%B1-a8c9c68748f6 adresinden alındı