MAKİNE ÖĞRENMESİ İLE SAĞLIK DURUMU TAHMİNLEME

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans Dönem Ödevi

EROL FAAL

Y210240114

Tez Danışmanı: PROF. DR. AYŞEGÜL ALAYBEYOĞLU SOY

Haziran 2023

İzmir Kâtip Çelebi Üniversitesi Fen Bilimleri Enstitüsü öğrencisi **Erol FAAL** tarafından hazırlanan **Makine Öğrenmesi İle Sağlık Durumu Tahminleme** başlıklı bu çalışma tarafımızca okunmuş olup, yapılan savunma sınavı sonucunda kapsam ve nitelik açısından başarılı bulunarak jürimiz tarafından YÜKSEK LİSANS ÖDEVİ olarak kabul edilmiştir.

ONAYLAYANLAR:

Tez Danışmanı: **Prof. Dr. Ayşegül Alaybeyoğlu SOY**

İzmir Kâtip Çelebi Üniversitesi

Savunma Tarihi: __.__.2023

Yazarlık Beyanı

Ben, **Erol FAAL**, başlığı **Makine Öğrenmesi ile Sağlık Durumu Tahminleme** olan bu tezimin ve tezin içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans / Doktora derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: 23.06.2023

Makine Öğrenmesi İle Sağlık Durumu Tahminleme

Öz

Makine Öğrenmesi ile Sağlık Durumu Tahminleme, spesifik olarak Kalp Hastalığı Tahmini projesi, yaşam tarzı ve sağlık parametrelerine dayalı olarak kardiyovasküler hastalık riskini değerlendirmek için makine öğrenimi tekniklerini kullanan web tabanlı bir uygulama geliştirmeyi amaçlamaktadır. Flask ve Scikit-learn entegrasyonuyla, proje kullanıcılara kişisel bilgilerini girmelerine ve kardiyovasküler risklerine yönelik kişiselleştirilmiş bir tahmin elde etmelerine imkan tanıyan oldukça basit bir arayüz sunar. Uygulama, cinsiyet, yaş, boy, kilo, kan basıncı, kolesterol seviyesi, glukoz seviyesi, sigara alışkanlığı, alkol tüketimi ve fiziksel aktivite gibi faktörleri dikkate alarak risk değerlendirmeleri yapar. HTML, CSS ve JavaScript kullanılarak geliştirilen web tabanlı arayüz, etkileşimli özellikleriyle kullanıcının bilgilerini girmesi konusunda yardımcı olur. Proje, teknolojilerin başarılı bir şekilde entegrasyonunu sergileyerek etkili bir kalp hastalığı tahmin uygulaması oluşturur. Gelecekteki geliştirmeler arasında tahmin modelinin doğruluğunun arttırılması, özellik setinin genişletilmesi ve veri görselleştirme tekniklerinin kullanılması yer alır. Sonuç olarak, Kalp Hastalığı Tahmini projesi, bireylere proaktif sağlık yönetimi ve önleyici önlemler konusunda değerli bir araç sunmaktadır.

Anahtar Sözcükler: Kalp Hastalığı Tahmini, Makine Öğrenimi, Web Uygulaması, Risk Değerlendirmesi, Yaşam Tarzı Faktörleri, Sağlık Parametreleri, Flask, scikit-learn, Kişiselleştirilmiş Tahmin, Kardiyovasküler Risk,

Health Prediction Using Machine Learning

Abstract

The Health Prediction Using Machine Learning, specifically Heart Disease Prediction project aims to develop a web-based application that utilizes machine learning techniques to assess the risk of cardiovascular disease based on lifestyle and health parameters. By integrating Flask and scikit-learn, the project delivers a user-friendly interface for users to input their personal information and obtain a personalized prediction of their cardiovascular risk. The application incorporates factors such as gender, age, height, weight, blood pressure, cholesterol level, glucose level, smoking habit, alcohol intake, and physical activity to generate accurate risk assessments. The web-based interface, developed using HTML, CSS, and JavaScript, ensures a seamless user experience with interactive features. The project showcases the successful integration of technologies in creating an efficient heart disease prediction application. Future enhancements include improving the prediction model's accuracy, expanding the feature set, and incorporating data visualization techniques. Overall, the Heart Disease Prediction project provides individuals with a valuable tool for proactive health management and preventive measures.

Keywords: Heart Disease Prediction, Machine Learning, Web Application, Risk Assessment, Lifestyle Factors, Health Parameters, Personalized Prediction, Cardiovascular Risk, Preventive Measures, Data Visualization.

Teşekkürler

- Proje danışmanım Sayın Profesör Ayşegül Alaybeyoğlu Soy,
- Ve bu programda gelişimime katkı sağlamış bütün hocalarıma gönülden teşekkür ederim.

İçindekiler

Bölüm 1

1.1 Projenin Konusu ve Amacı

Kalp hastalıkları, dünya genelinde ölüm nedenleri arasında önemli bir yer tutmaktadır. Özellikle kalp sağlığını etkileyen bir çok biyolojik ve sosyal etken bulunmakta olup, bireyin yaşam tarzı ve kalitesi ile kalp sağlığı arasında doğrudan bağlantılar kurulabilmektedir. Bu nedenle, bireylerin kendi sağlık durumlarını anlamaları ve potansiyel kardiyovasküler riskleri belirlemeleri hayati önem taşır. Ancak bu risklerin istatistiksel olarak değerlendirip, anlamlı bir sonuç ortaya çıkarmak oldukça sofistike bir çaba olacağından, uygun bir makine öğrenmesi modeli ile risk faktörlerinin karşılaştırılması ve tahminleme "Kalp Hastalığı Tahmini" projesi, bu ihtiyaca yönelik olarak geliştirilen bir web tabanlı uygulamadır.

Projenin konusu, kişilerin yaşam tarzı faktörlerini ve sağlık parametrelerini temel alarak kardiyovasküler hastalık risklerini tahmin etmektir. Bu faktörler arasında cinsiyet, yaş, boy, kilo, kan basıncı, kolesterol seviyesi, glukoz seviyesi, sigara alışkanlığı, alkol tüketimi ve fiziksel aktivite seviyesi yer almaktadır. Proje, bu parametreleri analiz etmek için makine öğrenimi algoritmalarını kullanır ve bireylerin kardiyovasküler risklerini objektif bir şekilde değerlendirir.

Projenin amacı, kullanıcı dostu bir arayüz aracılığıyla bireylere kişiselleştirilmiş bir tahmin sunmaktır. Kullanıcılar, web uygulaması üzerinden kendilerine ait sağlık verilerini girebilir ve kardiyovasküler risklerini değerlendirebilirler. Böylece, kişinin kendi sağlık durumunu daha iyi anlaması ve potansiyel risk faktörlerini fark etmesi sağlanır.

"Kalp Hastalığı Tahmini" projesi, bireylerin sağlık bilincini artırmayı hedefler. Uygulama, kullanıcılara sağlık konusunda bilinçli kararlar almaları için rehberlik

eder ve potansiyel risk faktörlerine dikkat etmelerini teşvik eder. Ayrıca, sağlıklı yaşam tarzı değişiklikleri ve önleyici önlemler konusunda bilgilendirme yapar. Bu sayede, bireylerin kendi sağlık yönetimlerini aktif bir şekilde yapmaları ve kalp hastalığı gibi ciddi sağlık sorunlarından korunmaları hedeflenir.

"Kalp Hastalığı Tahmini" projesi, sağlık bilincini artırmak ve erken tedbirler alarak kalp hastalıklarının yaygınlığını azaltmak için önemli bir adımdır. Kullanıcı dostu arayüzü, güvenilir tahminler ve kişiselleştirilmiş öneriler ile bireylerin kalp sağlığını iyileştirmelerine yardımcı olur.

1.2 Kaynakların Değerlendirilmesi

Bu proje için Kaggle'da bulunan kapsamlı bir kardiyovasküler hastalık dataseti seçilmiştir. Bu veri setinin seçilme sebebi, sağlık alanında önemli bir konu olan kardiyovasküler hastalıklar üzerine yoğunlaşan geniş ve güvenilir bir veri setine ihtiyaç duyulmasıydı.

Seçilen dataset, binlerce bireyin klinik ve demografik bilgilerini içermektedir. Yaş, cinsiyet, kan basıncı, kolesterol seviyeleri, kan glukozu, sigara ve alkol tüketimi gibi önemli parametreleri içeren veri seti, kardiyovasküler hastalık riskini değerlendirmek için ideal bir kaynaktır. Datasetin içerisinde kullanılan parametreleri ayrıntılı olarak açıklamak gerekirse:

1. **Age (Yaş)**: Bireyin yaşını temsil eder. Yaş, kardiyovasküler hastalıkların risk değerlendirmesinde önemli bir faktördür. Yaşın ilerlemesi, genellikle kardiyovasküler sorunlarla ilişkilendirilen riskleri artırır.
2. **Gender (Cinsiyet)**: Bireyin cinsiyetini ifade eder. Cinsiyet, kardiyovasküler hastalıklar üzerinde farklı etkilere sahip olabilir. Bazı faktörlerin, örneğin hormon seviyelerinin, cinsiyetle ilişkili olarak riski etkileme potansiyeli vardır.
3. **Height (Boy)**: Bireyin boyunu temsil eder. Boy, vücut kompozisyonu ve genel sağlık durumuyla ilişkili olabilir. Boyun normal aralıkta olması, sağlıklı bir vücut ağırlığının sürdürülmesi açısından önemlidir.

4. **Weight (Kilo)**: Bireyin kilosunu ifade eder. Vücut ağırlığı, obezite gibi sağlık sorunlarıyla ilişkilendirilen önemli bir faktördür. Sağlıklı bir kilonun korunması, kardiyovasküler hastalık riskini azaltmaya yardımcı olabilir.
5. **Ap_hi (Sistolik Kan Basıncı)**: Bireyin sistolik kan basıncını temsil eder. Sistolik kan basıncı, kalp kasının kasıldığı anki kan basıncını ifade eder. Yüksek sistolik kan basıncı, kardiyovasküler sorunlarla ilişkilendirilen riskleri artırabilir.
6. **Ap_lo (Diyastolik Kan Basıncı)**: Bireyin diyastolik kan basıncını temsil eder. Diyastolik kan basıncı, kalp kasının gevşediği anki kan basıncını ifade eder. Yüksek diyastolik kan basıncı da kardiyovasküler hastalık riskini artırabilir.
7. **Cholesterol (Kolesterol)**: Bireyin kolesterol seviyesini ifade eder. Yüksek kolesterol düzeyleri, kalp hastalığı riskini artırabilir. Kolesterol seviyesinin kontrol altında tutulması önemlidir.
8. **Gluc (Kan Glukozu)**: Bireyin kan glukoz seviyesini temsil eder. Yüksek kan glukoz seviyeleri, diyabet ve kardiyovasküler hastalık riskini artırabilir.
9. **Smoke (Sigara Kullanımı)**: Bireyin sigara kullanımını ifade eder. Sigara kullanımı, kardiyovasküler hastalıklar için önemli bir risk faktörüdür. Sigara içmek, damarları daraltarak kan basıncını yükseltebilir ve kalp sağlığını olumsuz etkileyebilir.
10. **Alco (Alkol Tüketimi)**: Bireyin alkol tüketimini temsil eder. Aşırı alkol tüketimi, kardiyovasküler sorunlarla ilişkilendirilen riskleri artırabilir. Alkol tüketimi kontrol altında tutulmalıdır.
11. **Active (Fiziksel Aktivite)**: Bireyin fiziksel aktivite düzeyini ifade eder. Düzenli fiziksel aktivite, kalp sağlığı için önemlidir. Aktif bir yaşam tarzı, kardiyovasküler hastalık riskini azaltabilir.
12. **Cardio (Kardiyovasküler Hastalık Durumu)**: Bu parametre, bireyin kardiyovasküler hastalık durumunu belirtir. Yani çıkış değerleridir. Modelin öğrenme sürecinde kullanılan hedef değişkenidir. Proje, bu hedef değişkenini tahmin etmeyi amaçlar ve bireylerin kardiyovasküler hastalık riskini değerlendirir.

Dataset, gerçek dünya verilerini yansıtması ve çeşitli parametrelerin etkileşimini içermesi açısından değerlidir. Bu da makine öğrenmesi algoritmalarını eğitirken daha doğru ve güvenilir sonuçlar elde etmeyi sağlar.

Kaggle üzerinden erişilebilen bu veri seti, projenin temelini oluşturarak, kardiyovasküler hastalık riskini tahmin etmek için geliştirilen algoritmanın etkinliğini ve doğruluğunu değerlendirmede büyük bir rol oynamıştır.

1.3 Kullanılan Teknolojiler

Projede kullanılan teknolojilerin ayrıntılı listesi:

1. **Python**: Python, basitliği ve okunabilirliği ile bilinen, çok yönlü ve yaygın olarak kullanılan bir programlama dilidir. Veri analizi, makine öğrenimi ve web geliştirme için zengin bir kütüphane ve çerçeve ekosistemine sahiptir. Projenin model geliştirme ve sunucu tarafı mantığını uygulamak için Python dili temel olarak kullanılmıştır.
2. **Flask**: Flask, Python için hafif ve esnek bir web frameworktür. Yönlendirme, HTTP istekleri ve template rendering gibi görevleri ele alarak web uygulamalarının oluşturulmasını sağlar. Bu projede, Flask, kullanıcıların tahmin modeliyle etkileşimde bulunabildiği web uygulamasını oluşturmak için kullanılmıştır.
3. **HTML/CSS**: HTML (Hypertext Markup Language) ve CSS (Cascading Style Sheets), sırasıyla web sayfalarının yapısını ve stilini oluşturmak için kullanılan standart dillerdir. HTML, kullanıcı arayüzünün öğelerini ve düzenini tanımlamaktan sorumludur, CSS ise web uygulamasının görsel sunumunu geliştirmek için kullanılır.
4. **JavaScript**: JavaScript, web sayfalarına etkileşim ve dinamik davranış eklemek için kullanılan yaygın bir programlama dilidir. Bu projede, JavaScript kullanıcı girişlerini yönetmek, istemci tarafında doğrulamalar

yapmak ve kullanıcı eylemlerine göre kullanıcı arayüzünü güncellemek için kullanılmıştır.

5. **Scikit-learn**: scikit-learn, Python için güçlü ve yaygın olarak kullanılan bir makine öğrenimi kütüphanesidir. Makine öğrenimi algoritmaları, ön işleme teknikleri ve değerlendirme metrikleri sunar. Bu projede, scikit-learn, sağlanan veri kümesini kullanarak tahmin modelini eğitmek ve kullanıcı girdisine dayalı tahminler yapmak için kullanılır.
6. **Pickle**: pickle, Python'ın içinde bulunan bir modüldür ve nesnelerin seri hale getirilmesi ve seriden çıkarılması için kullanılır. Bu projede, eğitilmiş makine öğrenimi modelini seri hale getirmek için pickle kullanılır. Bu, modelin diske kolayca kaydedilmesine ve ihtiyaç duyulduğunda yüklenmesine olanak tanır, verimli bir model sürekliliği sağlar.
7. **NumPy**: NumPy, Python'da bilimsel hesaplama için temel bir kütüphanedir. Büyük, çok boyutlu diziler ve matrisler için destek sağlar ve bu diziler üzerinde işlem yapmak için geniş bir matematiksel fonksiyon koleksiyonu sunar. Kalp Hastalığı Tahmini projesinde, NumPy, veri manipülasyonu, işleme ve makine öğrenimi modeli için giriş verilerini hazırlamak için kullanılır.
8. **SVC** (Support Vector Classifier): projede kullanılan bir makine öğrenimi algoritmasıdır. Scikit-learn kütüphanesinin bir parçasıdır ve sınıflandırma problemlerini çözmek için kullanılır.

   Projede, SVC algoritması kullanılarak bir kalp hastalığı tahmin modeli oluşturuldu. SVC, veri setindeki özellikleri analiz ederek ve belirli bir sınıfa ait etiketleri öngörerek kalp hastalığı olasılığını tahmin etmek için kullanıldı.

Bölüm 2

2.1 Makine Öğrenmesi Modeli Oluşturma

```
import pandas as pd
```

```
df = pd.read_csv("cardio_train.csv", sep=";")
print("Number of rows: %d" % df.shape[0])
print("Number of columns: %d" % df.shape[1])
```

```
Number of rows: 70000
Number of columns: 13
```

İlk satırda, pandas kütüphanesini pd kısaltmasıyla içe aktarıyoruz. İkinci satırda, **pd.read_csv()** fonksiyonunu kullanarak "**cardio_train.csv**" adlı CSV dosyasını okuyoruz. **sep=";"** parametresi, dosyanın sütunları ayırmak için kullanılan ayırıcı karakterin noktalı virgül olduğunu belirtir. Bu kod, dosyayı bir DataFrame olarak **df** değişkenine yükler.

Üçüncü satırda, **df.shape[0]** ifadesi, df veri çerçevesinin satır sayısını temsil eder. **df.shape[1]** ifadesi ise sütun sayısını temsil eder. Bu değerler **%d** formatlama belirteci kullanılarak ekrana yazdırılır.

Sonuç olarak, bu kod parçası, "**cardio_train.csv**" dosyasının içeriğini bir DataFrame olarak okur ve bu veri kümesinin kaç satır ve kaç sütundan oluştuğunu ekrana yazdırır.

```
print(df.head())
```

```
   id    age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  \
0   0  18393       2     168    62.0    110     80            1     1      0
1   1  20228       1     156    85.0    140     90            3     1      0
2   2  18857       1     165    64.0    130     70            3     1      0
3   3  17623       2     169    82.0    150    100            1     1      0
4   4  17474       1     156    56.0    100     60            1     1      0

   alco  active  cardio
0     0       1       0
1     0       1       1
2     0       0       1
3     0       1       1
4     0       0       0
```

Datasetin ilk 5 satırını inceliyoruz. Görüldüğü gibi yaş kısmı yıl değil gün olarak girilmiş. Ayrıca id kolonu bize istatistiksel açıdan anlamlı bir veri sunmadığı için değerlendirme dışına alabiliriz.

```
print(df.isnull().sum())
```

```
id             0
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
dtype: int64
```

```
#The age column offers only values in days. Let's convert them into years. We will convert them into even years for easer application:

df['age'] = round(df['age']/365.25,2)
```

```
#We drop the id column for it does not offer us significance for our analysis:

df.drop(['id'], axis=1, inplace=True)
```

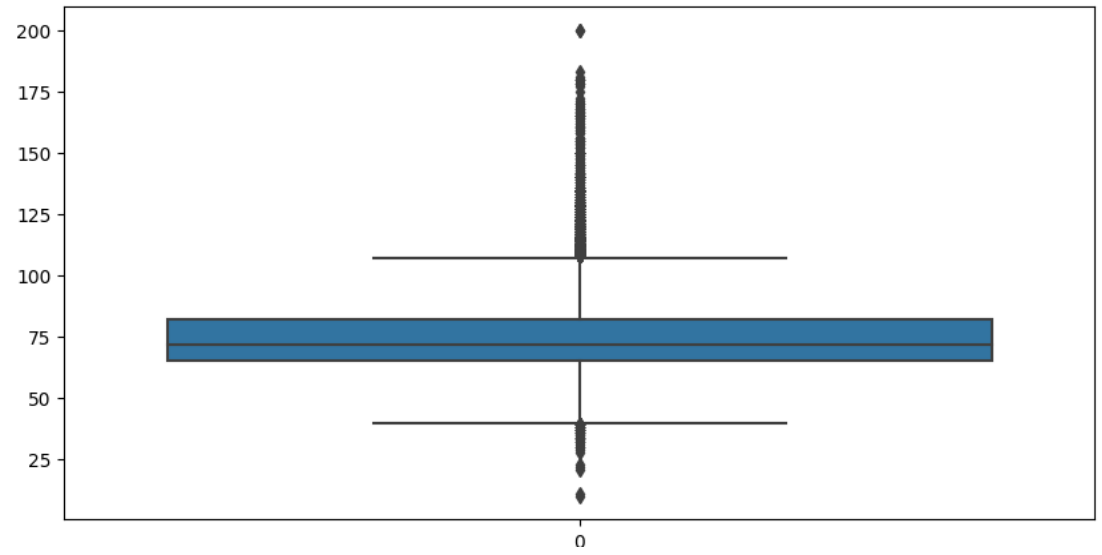
Gerekli değişiklikleri yaptıktan sonra tekrar verisetini incelediğimizde yaş kolonunun yıl temelli olduğundan daha rahat değerlendirilmesi ve id kolonunun gitmesi ile gereksiz bilgilerin ayıklandığı görülüyor.

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	50.36	2	168	62.0	110	80	1	1	0	0	1	0
1	55.38	1	156	85.0	140	90	3	1	0	0	1	1
2	51.63	1	165	64.0	130	70	3	1	0	0	0	1
3	48.25	2	169	82.0	150	100	1	1	0	0	1	1
4	47.84	1	156	56.0	100	60	1	1	0	0	0	0

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	53.302869	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0.053771	0.803729	0.499700
std	6.754974	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0.225568	0.397179	0.500003
min	29.560000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	48.360000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
50%	53.940000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
75%	58.390000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.000000	1.000000	1.000000
max	64.920000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.000000	1.000000	1.000000

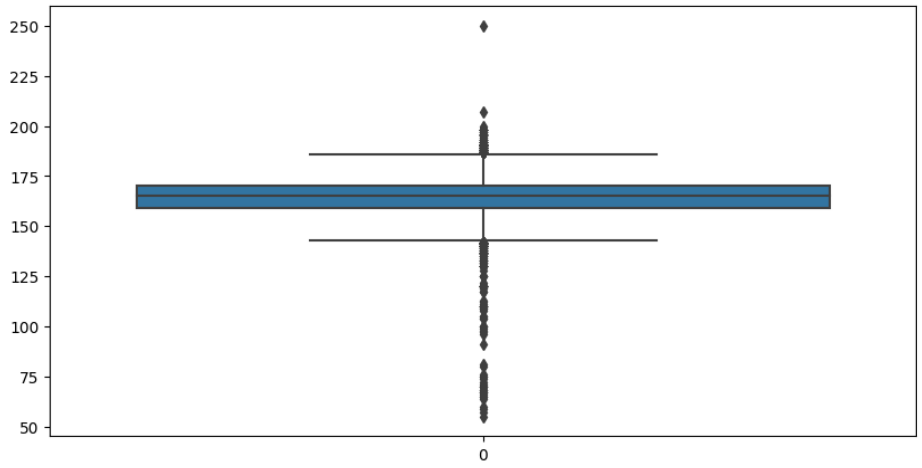
Ardından ortalamalar, minimum ve maksimum büyüklükler gibi temel istatistiksel veriler incelenebiliyor. Görüldüğü üzere bir çok kolonda bir çok outlier bulunuyor. Bu da makine öğrenmesi algoritmasının performansını bozma riskini de beraberinde getiriyor.

```
#It can be clearly seen that these outliers my affect the results drammatically
outlier_fig = plt.figure(figsize=(10,5))
sns.boxplot(df.weight)
plt.xlabel = ('Distribution of Weight')
plt.show()
```



Özellikle kilo kolonundaki verinin boxplot ile dağılımı incelendiğinde ciddi bir outlier içeriğinin mevcut olduğu görülüyor. Aşağıdaki grafikte ise boy verilerinin dağılımındaki dengesizlik açıkça görülüyor:

```
outlier_fig = plt.figure(figsize=(10,5))
sns.boxplot(df.height)
plt.xlabel = ('Distribution of Height')
plt.show()
```




```
#We determine what is the percentage of the uppermost and lowermost values of height, weight, sistolic and diastolic pressures.
(df['height'] > df['height'].quantile(0.975)).sum()/len(df.index)
```

```
0.020557142857142857
```

```
(df['height'] < df['height'].quantile(0.025)).sum()/len(df.index)
```

```
0.021957142857142856
```

```
(df['weight'] > df['weight'].quantile(0.975)).sum()/len(df.index)
```

```
0.023042857142857143
```

```
(df['weight'] < df['weight'].quantile(0.025)).sum()/len(df.index)
```

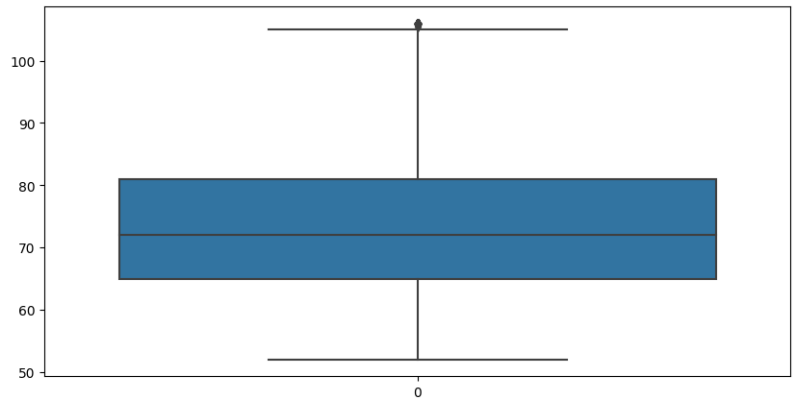
```
0.021285714285714286
```

Dataset içerisindeki outlier verilerin toplam içerisindeki oranları belirlenir. Böylece datasetin kapsamının ne kadar değişeceği değerlendirilmesi yapılabilir. Veriler gösteriyor ki outlierlar sonucu olan kayıp kabul edilebilir düzeyde.

```
#The percentage of outliers in our dataset is not too significant to keep. We can drop them:

df.drop(df[(df['height'] > df['height'].quantile(0.975)) | (df['height'] < df['height'].quantile(0.025))].index,inplace=True)
df.drop(df[(df['weight'] > df['weight'].quantile(0.975)) | (df['weight'] < df['weight'].quantile(0.025))].index,inplace=True)
```

Kilo ve boy kolonlarındaki alt ve üst **%2.5**'lik değerler silinir. Bu sayede kilo değerleri görüldüğü gibi daha dengeli bir dağılım göstermiştir.

```
#Revised box plot shows a more homogeneous distribution:

outlier_fig = plt.figure(figsize=(10,5))
sns.boxplot(df.weight)
plt.xlabel = ('Distribution of Weight')
plt.show()
```




```
#We apply the same principle to the values of pressure values

(df['ap_hi'] > df['ap_hi'].quantile(0.975)).sum()/len(df.index)
```

```
0.02498982244073529
```

```
(df['ap_hi'] < df['ap_hi'].quantile(0.025)).sum()/len(df.index)
```

```
0.01761500641969123
```

+ Code + Text

```
(df['ap_lo'] > df['ap_lo'].quantile(0.975)).sum()/len(df.index)
```

```
0.02317351955657157
```

```
(df['ap_lo'] < df['ap_lo'].quantile(0.025)).sum()/len(df.index)
```

```
0.0022390630382363073
```

Aynı temizleme stratejisini sistolik ve diyastolik basınç kolonlarında da uygulamamız gerekir.

```
#The percentage of outliers in the relevant columns is not too significant to keep as well. We can drop them:

df.drop(df[(df['ap_hi'] > df['ap_hi'].quantile(0.975)) | (df['ap_hi'] < df['ap_hi'].quantile(0.025))].index,inplace=True)
df.drop(df[(df['ap_lo'] > df['ap_lo'].quantile(0.975)) | (df['ap_lo'] < df['ap_lo'].quantile(0.025))].index,inplace=True)
```

Aşağıdaki özete göre artık daha dengeli dağılım gösteren bir datasetimiz olduğunu **describe()** methodu ile görebiliyoruz:

```
df.describe()
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
count	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000	60142.000000
mean	53.302468	1.347311	164.554854	73.426805	125.770526	81.046307	1.350953	1.220229	0.085631	0.051877	0.803648	0.488228
std	6.736515	0.476120	6.830174	11.614806	13.761847	8.239157	0.670076	0.567607	0.279820	0.221781	0.397241	0.499866
min	29.560000	1.000000	150.000000	52.000000	100.000000	60.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	48.400000	1.000000	160.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
50%	53.950000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
75%	58.370000	2.000000	169.000000	80.000000	135.000000	90.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000
max	64.920000	2.000000	180.000000	106.000000	163.000000	100.000000	3.000000	3.000000	1.000000	1.000000	1.000000	1.000000

Yukarıdaki tabloda kolonlar arası korelasyon haritasını görebiliyoruz.

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, cross_validate
import warnings
warnings.filterwarnings('ignore')
from sklearn.svm import SVC, LinearSVC
import os
```

Import kodlarımız, scikit-learn kütüphanesinden bazı modülleri içe aktarır ve warnings kütüphanesini kullanarak uyarı mesajlarını görmezden gelir. Ayrıca, os kütüphanesini de içe aktarır. Kullanılan modüllerin açıklamaları:

sklearn.metrics.accuracy_score: Bu modül, sınıflandırma problemlerinde doğruluk (accuracy) skorunu hesaplamak için kullanılır. Gerçek ve tahmin edilen sınıfların karşılaştırılmasıyla doğruluk skoru elde edilir.

sklearn.model_selection.train_test_split: Bu modül, dataseti eğitim ve test kümelerine ayırmak için kullanılır. Dataseti, belirtilen oranlarda rastgele bir şekilde bölünür.

sklearn.model_selection.cross_val_score: Bu modül, cross-validation kullanarak bir modelin performansını değerlendirmek için kullanılır. Veri kümesi, belirli bir sayıda parçaya bölünerek her bir parçada model eğitilir ve diğer parçalarda test edilir.

sklearn.model_selection.GridSearchCV: Bu modül, hiperparametre ayarlaması yapmak için kullanılır. GridSearchCV, belirli bir hiperparametre uzayını araştırarak en iyi hiperparametre kombinasyonunu bulmaya çalışır.

sklearn.svm.SVC ve **sklearn.svm.LinearSVC**: Bu modüller, SVM algoritmalarını uygulamak için kullanılır. SVC modülü, çekirdek işlevi kullanarak SVM'yi uygularken, LinearSVC modülü doğrusal SVM'yi uygular.

os: Bu modül, işletim sistemine bağımlı işlemler yapmak için kullanılır. Örneğin, dosya yollarını oluşturma veya silme gibi işlemler gerçekleştirmek için kullanılabilir.

```
X = df.drop(columns=["cardio"])
y = df["cardio"]

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Bu kod , dataseti bağımsız değişkenler **(X)** ve hedef değişken **(y)** olarak ayırır. Ardından, veriyi eğitim ve test kümelerine böler.

X = df.drop(columns=["cardio"]): df veri çerçevesinden "cardio" sütununu çıkararak bağımsız değişkenleri (X) oluşturur. "Cardio" sütunu, hedef değişkeni temsil eder ve bu sütunu çıkararak geri kalan sütunları bağımsız değişkenler olarak kullanılır.

y = df["cardio"]: df veri çerçevesinin "**cardio**" sütununu hedef değişken (y) olarak ayırır. "cardio" sütunu, tahmin etmeye çalıştığımız sınıf etiketlerini içerir.

train_test_split(X, y, test_size=0.3): train_test_split fonksiyonunu kullanarak veriyi eğitim ve test kümelerine böler. X bağımsız değişkenlerini ve y hedef değişkenini vererek, veriyi rastgele bir şekilde eğitim ve test kümelerine böler. test_size=0.3 parametresi, verinin %30'unun test kümesine ayrılacağını belirtir. Bu oran isteğe bağlı olarak değiştirilebilir.

Sonuç olarak, **X_train, X_test, y_train** ve **y_test** değişkenleri, datasetimizi eğitim ve test kümelerine bölmek için kullanılır. **X_train** ve **y_train**, modelin eğitimi için kullanılacak eğitim verilerini temsil ederken, **X_test** ve **y_test**, modelin performansını değerlendirmek için kullanılacak test verilerini temsil eder.

```
svc_model = SVC(C=100, gamma=0.00001, kernel="rbf", random_state=42)
svc_cv = cross_validate(svc_model, X, y, cv=3)
svc_cv
```

```
{'fit_time': array([65.5212121 , 65.91988778, 65.36640453]),
 'score_time': array([32.32260537, 32.95724297, 32.40728092]),
 'test_score': array([0.72316441, 0.7199581 , 0.71950915])}
```

Birçok makine öğrenmesi algoritması denemiş olup, test skoru en yüksek olan SVC algoritması seçilmiştir. Modelin oluşturulduğu değerleri:

svc_model = SVC(C=100, gamma=0.00001, kernel="rbf", random_state=42): SVC modelini oluştururken, C, gamma, kernel ve random_state gibi hiperparametreleri belirlenir. C hiperparametresi, C değeri büyüdükçe modelin daha fazla eğitim örneğine uymaya çalışacağını belirleyen bir düzenleme parametresidir. gamma hiperparametresi, çekirdek fonksiyonunun genişliğini kontrol eder. kernel hiperparametresi, SVC'nin kullanacağı çekirdek fonksiyonunu belirler (burada "rbf" olarak belirtilmiştir, yani RBF çekirdeği). random_state hiperparametresi ise rastgelelik durumunu kontrol eder, böylece sonuçlar tekrarlanabilir hale gelir.

svc_cv = cross_validate(svc_model, X, y, cv=3): cross_validate fonksiyonunu kullanarak çapraz doğrulama işlemi gerçekleştirilir. svc_model parametresi, değerlendirilecek modeli temsil eder. **X** ve **y** ise bağımsız değişkenler ve hedef değişkeni içeren veri setini temsil eder. **cv**=3 parametresi, veri setini 3 parçaya bölmek için 3 kat çapraz doğrulama yapılacağını belirtir.

```
score
```

```
0.7167322507343569
```

Bu skor, modelin doğruluk oranını yani doğru sınıflandırma oranını gösterir. Doğruluk, doğru olarak sınıflandırılan örneklerin toplam örnek sayısına oranıdır. Örneğin, 0.7167 doğruluk skoru, modelin test veri setindeki örneklerin yaklaşık %71.67'sini doğru şekilde sınıflandırdığını gösterir.

```
import pickle
pickle_out = open("classifier2.pkl","wb")
pickle.dump(svc_model, pickle_out)
#pickle_out.close()
```

```
model = pickle.load(open("classifier2.pkl","rb"))

sample = [[57, 1, 180, 150, 125, 81, 1, 2, 1, 1, 0]]

model.predict([[57, 1, 180, 150, 125, 81, 1, 2, 1, 1, 0]])
```

Pickle, Python programlama dilinde nesnelerin pickling ve unpickling için kullanılan bir modüldür. Pickle, Python nesnelerini baytlara dönüştürmek ve bu baytları bir dosyaya veya ağ üzerinden iletmek için kullanılır. Pickling işlemi, bir nesnenin bellekteki durumunu koruyarak onu bir veri akışı veya dosya üzerinde taşınabilir hale getirir. Unpickling işlemi ise pickling olmuş veriyi okuyarak tekrar Python nesnesine dönüştürür.

Pickle, bir nesnenin tam durumunu koruyabilen bir serileştirme yöntemidir. Bu, nesnelerin hiyerarşik yapısını, içerdikleri verileri ve hatta nesnelerin kendi yöntemlerini (metodlarını) koruyabildiği anlamına gelir. Pickle, veri analizi, model eğitimi ve depolama gibi birçok alanda kullanışlıdır. Özellikle makine öğrenimi modellerini kaydetmek ve yeniden kullanmak için sıklıkla tercih edilir. Böylece tekrar model elde etmek için zaman ve enerji tüketimi gereğini ortadan kaldırır.

Yukarıdaki kodda görüldüğü gibi, öncelikle bir pickle dosyası elde edilir. Ardından sample adlı bir değişkene örnek parametreler yüklenerek pickle modelinin çalışma durumu test edilir. Bu durumda sonuç: `array([1])` olarak geri döner.

Böylece projemizin makine öğrenmesi model oluşturma bölümü tamamlanmış oluyor.

2.2 Modelin Back End Bölümünün Oluşturulması

app.py

```python
from flask import Flask, render_template, request
import pickle
import numpy as np

app = Flask(__name__)

# Load the pickled model
model = pickle.load(open('models/classifier2.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    # Get the form inputs
    gender = int(request.form['gender'])
    age = int(request.form['age'])
    height = int(request.form['height'])
    weight = int(request.form['weight'])
    hi = int(request.form['hi'])
    lo = int(request.form['lo'])
    cholesterol = int(request.form['cholesterol'])
    glucogen = int(request.form['glucogen'])
    smoke = int(request.form['smoke'])
    alcohol = int(request.form['alcohol'])
    activity = int(request.form['activity'])

    # Prepare the input data for prediction
    input_data = np.array([[gender, age, height, weight, hi, lo, cholesterol, glucogen, smoke, alcohol, activity]])

    # Make the prediction
    prediction = model.predict(input_data)[0]

    message_1 = "You have a high risk of having a cardiovascular condition. Please refer to a health professional as
    message_0 = "It appears you are in good health and do not seem to have a heart condition. I bid you a healthy li
    if prediction == 1:
        return render_template('index.html', prediction=message_1)
    else:
        return render_template('index.html', prediction=message_0)

if __name__ == '__main__':
    app.run(debug=True)
```

Bu kod seti, Flask web framework'ünü kullanarak bir web uygulaması oluşturur. Uygulama, kullanıcının bir formu doldurduktan sonra eğitilmiş modelimizi kullanarak kardiyovasküler hastalık riskini tahmin eder ve sonucu kullanıcıya gösterir. Değişkenlerin ve fonksiyonların detaylı açıklaması:

app = Flask(__name__): Flask uygulamasını oluşturur.

model = pickle.load(open('models/classifier2.pkl', 'rb')): Serileştirilmiş modeli (classifier2.pkl) diskten yükler.

@app.route('/'): Ana sayfa için bir route tanımlar.

def home():: Ana sayfayı temsil eden bir fonksiyon tanımlar.

@app.route('/predict', methods=['POST']): Tahmin yapmak için bir route tanımlar ve bu route'a sadece POST istekleriyle erişilebileceğini belirtir.

def predict():: Tahmin yapmayı sağlayan bir fonksiyon tanımlar. Form girişlerini alarak kullanıcının girdiğini değerleri alır. Girdi verilerini tahmin yapmak için uygun formatta bir diziye dönüştürür. Modeli kullanarak tahmin yapar. Tahmin sonucuna göre bir mesaj oluşturur ve tahmin sonucunu ve mesajı ana sayfada gösterir.

if __name__ == '__main__':: Uygulama doğrudan çalıştırıldığında (import edilmediğinde) Flask uygulamasını başlatır.

app.run(debug=True): Uygulamayı debug modunda çalıştırır.

Bu applikasyon sayesinde, kullanıcının web tarayıcısı aracılığıyla bir form doldurup gönderdiği veriler makine öğrenmesi ile oluşturulmuş pickle dosyası ile değerlendirilir ve kardiyovasküler hastalık riskini tahmini yapılır. Tahmin sonucunu kullanıcıya geri döndürür ve web sayfasında gösterir.

2.3 Modelin Front End Bölümünün Oluşturulması

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Heart Disease Prediction</title>
    <style>
        .slider {
          width: 300px;
        }
      </style>
</head>
<body>
    <h2>Erol Faal - Katip Celebi University Software Eng. Master Project</h2><br>
  <p>Disclaimer: The following health prediction study may not be medically
    accurate and in no way should be taken as a medical advice.
    The values and results below are only for individual study purposes.

      <br><br>
    The scenario: Cardiovascular systems of humans play an essential role
    in general health. That is why many scientists have been closely
    studying what affects the health of a hearth. For that reason,
    we are going to use some variables that reflect individuals'
    lifestyle and predict the risks of them on one's cardiovascular system.
    That way we can present a more accurate information to those
    who would like to improve their health.</p>

    <h2>Please enter your respective values</h2>


    <form id="mlForm" action="/predict" method="post">

        <label for="gender">Gender:</label><br>
        <select id="gender" name="gender">
          <option value="0">Female</option>
          <option value="1">Male</option>
        </select>
        <br>

        <label for="age">Age:</label><br>
        <input type="range" id="age" name="age" min="0" max="100" value="50" step="1" class="slider">
        <span id="ageValue">50</span>
        <br>

        <label for="height">Height:</label><br>
        <input type="range" id="height" name="height" min="150" max="200" value="175" step="1" class="slider">
        <span id="heightValue">175</span>
```

```
 <label for="activity">Physical Activity:</label><br>
 <select id="activity" name="activity">
   <option value="0">Inactive</option>
   <option value="1">Active</option>
 </select>
 <br>

 <input type="submit" value="Submit">
 <input type="button" value="Reset" onclick="resetForm()">
/form>

h2>Prediction Result</h2>
p id="predictionResult"></p>

script>
 var ageSlider = document.getElementById("age");
 var ageValue = document.getElementById("ageValue");

 var heightSlider = document.getElementById("height");
 var heightValue = document.getElementById("heightValue");

 var weightSlider = document.getElementById("weight");
 var weightValue = document.getElementById("weightValue");

 var hiSlider = document.getElementById("hi");
 var hiValue = document.getElementById("hiValue");

 var loSlider = document.getElementById("lo");
 var loValue = document.getElementById("loValue");



 // Display the initial values of the sliders
 ageValue.innerHTML = ageSlider.value;
 heightValue.innerHTML = heightSlider.value;
 weightValue.innerHTML = weightSlider.value;
 hiValue.innerHTML = hiSlider.value;
 loValue.innerHTML = loSlider.value;


 // Update the values displayed beside the sliders when the sliders' values change
 ageSlider.oninput = function() {
   ageValue.innerHTML = this.value;
 };

 heightSlider.oninput = function() {
   heightValue.innerHTML = this.value;
 }.
```

Front End için kullanılan HTML, CSS ve Javascript kodlarından bazıları dikkatlere sdunulmuştur. Flask web uygulamasında kullanılmak üzere bir kullanıcı arayüzü oluşturur. Kullanıcıdan kardiyovasküler hastalık riskini tahmin etmek için giriş verilerini alır ve sonucu görüntüler.

Form alanları ve giriş verileri almak için bir HTML formu tanımlanmıştır. Bu form, kullanıcının cinsiyet, yaş, boy, kilo, kan basıncı, kolesterol düzeyi, glukoz düzeyi,

sigara içme durumu, alkol tüketimi ve fiziksel aktivite gibi bilgileri girmesine olanak tanır.

<input type="submit" value="Submit">: Formun gönderme düğmesi. Formdaki verileri sunucuya göndermek için kullanılır.

<input type="button" value="Reset" onclick="resetForm()">: Formu sıfırlamak için bir düğme. Bu düğmeye tıklandığında, formdaki giriş verileri sıfırlanır.

** :** Tahmin sonucunun görüntülendiği bir paragraf öğesi. Tahmin sonucu Flask uygulaması tarafından dinamik olarak bu paragrafın içine yerleştirilecektir.

JavaScript kodu, sayfadaki kaydırma çubuklarının değerlerini yakalar ve kullanıcının girişlerini gösterir. Ayrıca, tahmin sonucunun görüntülendiği paragrafın içeriğini günceller.

Flask uygulaması tarafından gönderilen **{{ prediction }}** değişkeni, JavaScript kodu tarafından tahmin sonucunun yerleştirileceği yerdir.

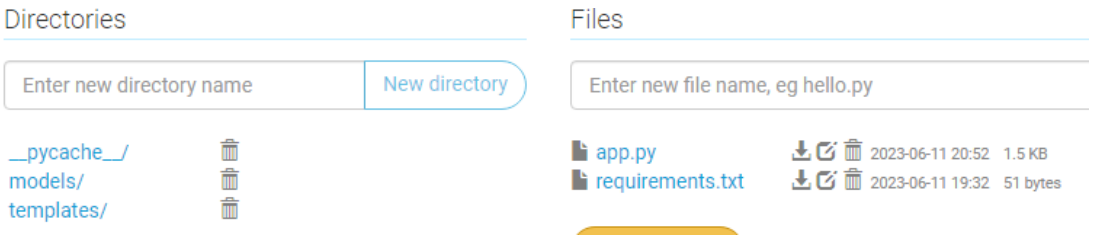
resetForm() işlevi, formu sıfırlar ve tahmin sonucunu temizler.

Bu HTML şablonu, Flask uygulamasıyla birlikte çalışarak kullanıcının giriş verilerini alır, tahmin yapar ve sonucu gösterir.

Bölüm 3

3.1 Web Uygulamasının Deploy ve Test Edilmesi.

Python kullanılarak oluşturulan bu web uygulamasının deploy edilmesi için PythonAnywhere (https://www.pythonanywhere.com/) sunucuları kullanılmıştır. Temel paketinin ücretsiz olması ve kullanım kolaylığı açısından tercih edilmiştir. Bu server ile öncelikle gereken kaynak dosyalar upload edilir:

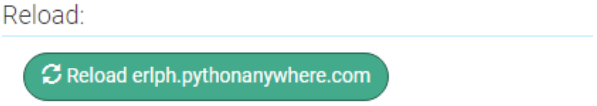


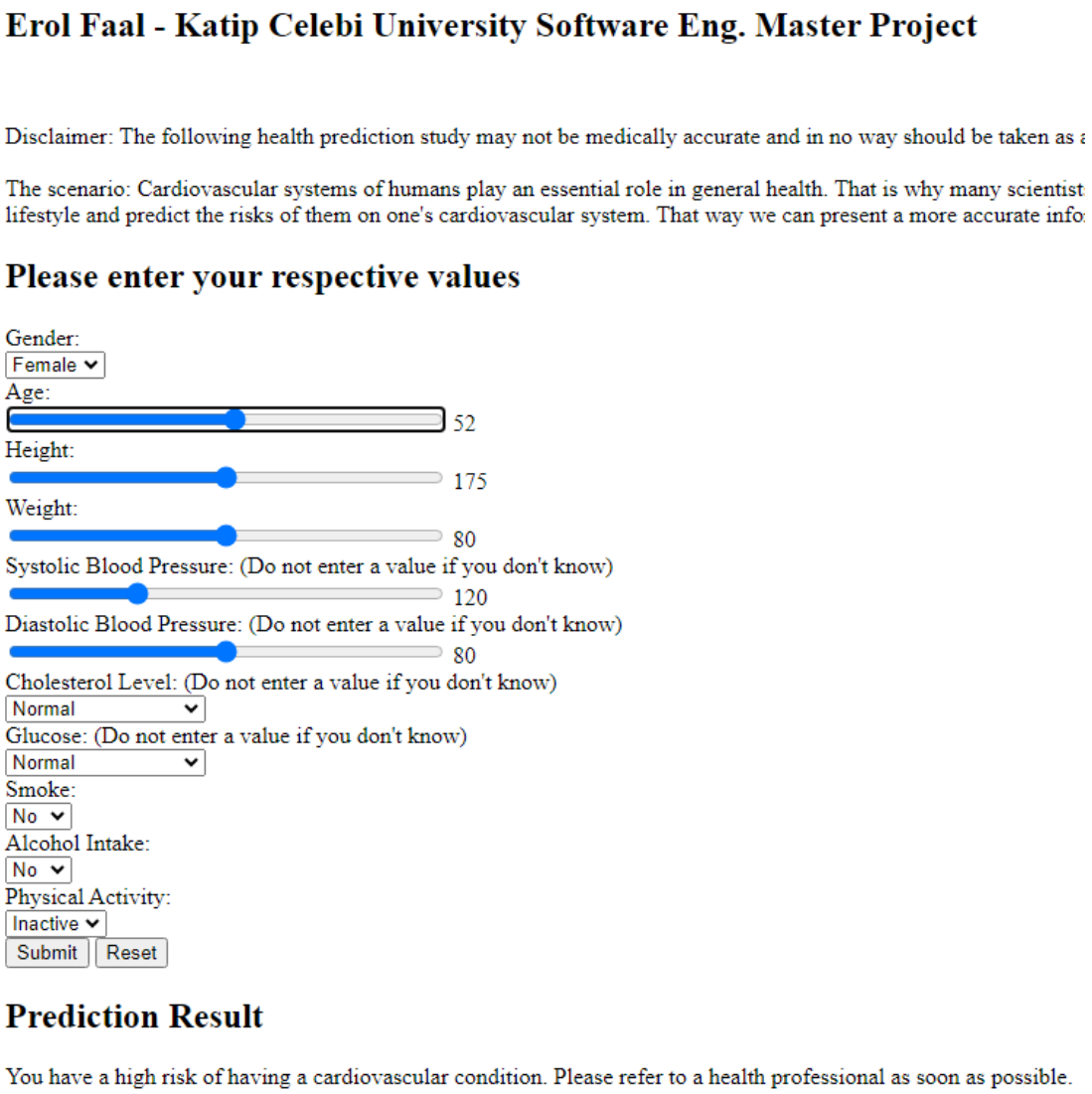
Ardından upload edilen uygulamanın nasıl çalıştırılacağına dair WSGİ (Web Server Gateway Interface) üzerine gerekli kodlar eklenir:

/var/www/erlph_pythonanywhere_com_wsgi.py

```
# This file contains the WSGI configuration required to serve up your
# web application at http://<your-username>.pythonanywhere.com/
# It works by setting the variable 'application' to a WSGI handler of s
# description.
#
# The below has been auto-generated for your Flask project

import sys

# add your project directory to the sys.path
project_home = '/home/erlph/1'
if project_home not in sys.path:
    sys.path = [project_home] + sys.path

# import flask app but need to call it "application" for WSGI to work
from app import app as application  # noqa
```

Uygulamanın serverda çalıştırılması:





http://erlph.pythonanywhere.com/ internet sitesinden ulaşabilen projenin son hali ve tahminleme sonrası mesajı yukarıdaki görüntüde sunulmuştur. Arayüz olabildiğince basit ve kullanıcı dostu olarak dizayn edilmiştir.

3.2 Sonuç

Projemiz, bir web tabanlı sağlık tahmin uygulamasının geliştirilmesi üzerine odaklanmaktadır. Uygulama, kullanıcıların kişisel sağlık verilerini girmelerine ve kardiyovasküler bir duruma sahip olma riskini tahmin etmelerine olanak tanır. Kullanıcılar cinsiyet, yaş, boy, kilo, kan basıncı, kolesterol seviyesi, glukoz düzeyi, sigara kullanımı, alkol tüketimi ve fiziksel aktivite gibi parametreler girer. Makine öğrenmesi modeli, kullanıcı girdilerini işleyerek kardiyovasküler durum riskini tahmin eder.

Bu proje, bir SVC (Support Vector Classifier) sınıflandırma modeli kullanılarak gerçekleştirilmiştir. Model, eğitim verileri üzerinde eğitilmiş ve ardından test verileriyle doğruluk değerlendirmesi yapılmıştır. Elde edilen sonuçlar, modelin yüksek sayılabilecek bir doğruluk oranı elde ettiğini göstermiştir.

Web uygulaması, Flask framework kullanılarak geliştirilmiş ve Pythonanywhere platformu üzerinde dağıtılmıştır. Kullanıcılar, uygulama aracılığıyla kişisel sağlık verilerini girerek kardiyovasküler durumları hakkında bilgi edinebilirler. Uygulama, kullanıcılara riskli bir kardiyovasküler duruma sahip olup olmadıklarını bildirir ve gerektiğinde bir sağlık uzmanına başvurmalarını önerir.

Bu proje, sağlık tahmininde makine öğrenmesi ve web teknolojilerinin birleşimini kullanarak kullanıcılara değerli bir hizmet sunmayı hedeflemektedir. Kullanıcıların sağlık durumlarını izlemelerine ve potansiyel sağlık risklerini erken tespit etmelerine yardımcı olmayı amaçlamaktadır.

Kaynaklar

1. Kaggle Cardio_train dataseti: https://www.kaggle.com/datasets/pirogovskiy/cardio-train
2. Scikit-learn Dokümantasyonu: https://scikit-learn.org
3. Flask Dokümantasyonu: https://flask.palletsprojects.com
4. Pythonanywhere: https://www.pythoneverywhere.com

Projede kullanılan bütün kaynak kodları aşağıdaki link ile erişilebilen Google Drive dosyası ile herkese açıktır:

https://drive.google.com/drive/folders/1Na6USsTBFYUtwMN1JD-jR8zmQ0Rck3N5?usp=sharing

Özgeçmiş

Adı Soyadı: Erol FAAL

Eğitim:

2012–2016	İzmir Yaşar Üniversitesi İngiliz Dili ve Edebiyatı
2013–2016	İzmir Yaşar Üniversitesi Aktüerya Bilimleri (Çift Anadal)

İş Deneyimi:

2019 – 2023 (Devam)	MEB - İngilizce Öğretmeni