Patient Survival Prediction

Feyza Şahin
18011019
Bilgisayar Mühendisliği
Yıldız Teknik Üniversitesi
İstanbul, Türkiye

Elif Ayanoğlu
19011089
Bilgisayar Mühendisliği
Yıldız Teknik Üniversitesi
İstanbul, Türkiye

***Abstract*—Günümüzde Makine Öğrenmesi teknikleriyle çeşitli sınıflandırma ve tahmin modelleri gerçekleştirilebilmektedir. Logistic Regression, Stochastic Gradient Descent, Support Vector Machine gibi modeller tahminleyici sistemler için kullanılan sınıflandırıcı modellerden birkaçıdır. Bu çalışmada da, hastaneye yatırılan hastaların özelliklerine bakılarak hastane içi mortalite oranı için bir tahmin modeli gerçekleştirilmiştir. Toplanan veri bu modellerde işlenmeden önce Veri Madenciliği yöntemleriyle temizlendikten sonra bu modellere eğitim ve test için verilmiştir. Çıkan sonuçlar ve özellikler python kütüphaneleri kullanılarak görselleştirilmiştir.**

***Index Terms*—logistic regression, stochastic gradient descent, support vector machine, prediction**

I. Introduction

Patient Survival Prediction adlı veri seti, hastaneye yatırılan hastalar arasında tüm nedenlere bağlı hastane içi mortalite yani ölüm oranı için bir tahmin modeli geliştirmeyi amaçlamaktadır. Veri setinde özellik olarak cinsiyet, yaş, ırk, vücut kitle indeksi, nabız değeri, solunum hızı, vücut sıcaklığı, apache değerleri, koma durumu için göz hareket durum bilgisi, hastanın transfer olup olmadığı gibi yaklaşık 80 durumun bilgisi verilmektedir. Bu çalışmanın amacı, bu verileri referans alarak yine bu özellikleri verilen bir hasta için durumunun ölümcül olup olmadığını tahmin eden modeller geliştirmektir. Kullanılan modeller Support Vector Machine, Lojistik Regresyon ve Stochastic Gradient Descent modelleridir. Gerçekleştirilen eğitimlerin başarım skorları tablolar ile gösterilmiştir. Projenin bütün kodlarına aşağıdaki github linkinden ulaşabilirsiniz: https://github.com/feysahin/Data-Mining

II. Preprocessing

Toplanan veriyi daha etkin kullanabilmek için bazı temizlemelerin yapılması gerekmektedir. Bu temizliklere 'null' değerlerin doldurulması, kategorik değerlerin encode edilerek numerik değerlere dönüştürülmesi, etkisi olmayan özelliklerden veri setinin arındırılması gibi işlemler örnek olarak verilebilir.

A. Verisetindeki ölüme etki etmeyen özelliklerin silinmesi:

Hazırlık aşamasında ilk olarak verisetimizde bulunan hasta kodu, hastane kodu gibi özellik olarak kullanamayacağımız bazı sütunları çıkardık.

```
df.drop("hospital_id", axis=1, inplace=True)
df.drop("patient_id", axis=1, inplace=True)
df.drop("encounter_id", axis=1, inplace=True)
df.drop("Unnamed: 83", axis=1, inplace=True)
df
```

Fig. 1. Çıkartılan sütunlar

B. Label Encoding:

Sütun temizliğinden sonra sıra string tipindeki özelliklerin sayısal karşılıklarına dönüştürme işlemine gelmektedir. Makineler modelleri sayısal değerlerle işlediğinden string tipindeki sütunların encode edilmesine ihtiyaç vardır. Sklearn kütüphanesinin LabelEncoder() fonksiyonu, parametre olarak aldığı sütunu, içerisindeki unique değerlere farklı bir sayısal karşılık vererek bu dönüşümü gerçekleştirir. Şekildeki fonksiyonumuz içerisinde string tipindeki her sütunumuz için bu işlemi gerçekleştirmekteyiz.

```
def encode_df(df):
    le = LabelEncoder()
    for column in df.columns:
        if df[column].dtypes == np.object:
            df[column] = le.fit_transform(df[column])
    return df
```

```
df = encode_df(df)
df
```

Fig. 2. Label Encoding işlemi

C. Boşluk değerleri doldurma:

Özelliklerin bazı örnekler için uygulanabilir olmaması, hatalı veri toplama yöntemleri ve süreçleri ya da yazılımsal-donanımsal nedenlerden dolayı verisetlerinde 'null' değerler bulunabilmektedir. 'null' değerlere sahip örnekleri silerek veri kaybı yaşamamak adına bu değerleri doldurmak için veri madenciliğinde çeşitli yöntemler mevcuttur. En yaygın yöntemlerden biri ortalama değerle doldurma yöntemidir. Biz de çalışmamızda bu yöntemi kullandık. [1]

Boşlukları doldururken (Fig 6) verisetimizi ölüm teşhisi konanlar ve konmayanlar olmak üzere ikiye ayırdık. Bunu yapmamızın sebebi iki grubun o özellik için olan ortalama

```
df[['icu_admit_source', 'apache_2_bodysystem']][:10]
```

	icu_admit_source	apache_2_bodysystem
0	Floor	Cardiovascular
1	Floor	Respiratory
2	Accident & Emergency	Metabolic
3	Operating Room / Recovery	Cardiovascular
4	Accident & Emergency	Trauma
5	Accident & Emergency	Neurologic
6	Accident & Emergency	Respiratory
7	Accident & Emergency	Cardiovascular
8	Other Hospital	Cardiovascular
9	Accident & Emergency	Cardiovascular

Fig. 3. Label Encoding'den önce

```
df[['icu_admit_source', 'apache_2_bodysystem']][:10]
```

	icu_admit_source	apache_2_bodysystem
0	1	0
1	1	6
2	0	3
3	2	0
4	0	7
5	0	4
6	0	6
7	0	0
8	3	0
9	0	0

Fig. 4. Label Encoding'den sonra

```
print("# of missing values:", df.isnull().any(axis = 1).sum())
# of missing values: 34115
```

Fig. 5. Verisetimizdeki toplam 'null' değer sayımız

değerlerini kendi içerisinde hesaplayıp özellik değerlerinin dağılımını etkilememesi düşüncesidir. Şekildeki fonksiyonda gösterildiği üzere her sütun için önce label değeri 1 olanların, sonra da 0 olanların ortalama değerleri hesaplanıp kendi boşluklarına kendi ortalamaları gelecek şekilde güncellemeler yapılmıştır. [0]

D. Feature Selection - PCA - Korelasyon

Bu aşamaların sonrasında özelliklerimiz fazla sayıda olduğundan feature selection yapabilmek için çeşitli yöntemler denedik fakat yine özellik sayımızın fazla olması nedeninden dolayı donanımsal olarak bellek sorunlarıyla karşılaştık. Dolayısıyla hazır fonksiyonlar kullanmak yerine öncelikle bütün özelliklerimizin label sütunuyla olan korelasyon değerlerine bakmayı tercih ettik (Fig 7). Mutlak değer olarak en yüksek korelasyona sahip özelliğimizin bile korelasyon değerinin oldukça düşük olduğu sonucuyla karşılaştık.

Bu nedenle özellik sayımızı azaltmanın underfitting problemine yol açacağını düşünerek sütun elememeye karar verdik. Ölüm gibi ciddi bir teşhisin çok fazla etkeninin olması

```
def full_null(col_name):

    # riskli hastalar için
    df_1 = df[df['hospital_death'] == 1]

    df_bmi_1_avg = df_1[col_name].mean()
    df_bmi_1_avg

    df_1[col_name] = df_1[col_name].apply(lambda x: df_bmi_1_avg if pd.isna(x)

    # riskli hastalar için
    df_0 = df[df['hospital_death'] == 0]

    df_bmi_0_avg = df_0[col_name].mean()
    df_bmi_0_avg

    df_0[col_name] = df_0[col_name].apply(lambda x: df_bmi_0_avg if pd.isna(x)

    # concatenate
    frames = [df_0[col_name], df_1[col_name]]
    df_col = pd.concat(frames)
    return df_col
```

```
for col in df.columns:
    df[col] = full_null(col)
df
```

Fig. 6. Null değerleri doldurma

```
cols = df.columns
new_cols = []
correlations = []
for i in range(len(cols)):
    column_1 = df[cols[i]]
    column_2 = df["hospital_death"]
    correlation = column_1.corr(column_2)
    if abs(correlation) >= 0.25:
        new_cols.append(cols[i])
        correlations.append(correlation)
        print(cols[i] , ": " , correlation)
```

```
gcs_eyes_apache :  -0.26037275119026804
gcs_motor_apache :  -0.28244919978844524
apache_4a_hospital_death_prob :  0.31104269498111126
apache_4a_icu_death_prob :  0.28391302632818544
hospital_death :  1.0
```

Fig. 7. Korelasyon hesaplama

aslında beklenen bir durumdur. Çalışmamızın alanı sağlık olduğundan dolayı da diğer alanlara nazaran daha dikkatli olmamız gerektiğini düşünerek modellerimizin bütün parametreleri dikkate almasını tercih ettik.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
st_df = []
st = scaler.fit_transform(df)
pd.DataFrame(st)
i=0
for col in df.columns:
    st_df[col] = st[:,i]
    i+=1
pd.DataFrame(st_df)
```

Fig. 8. Modeller için Standard Scaler

III. MODELLER

A. SVM

SVM yani Destek Vektör Makineleri, iki sınıfı bir doğru ya da düzlem yardımıyla birbirinden farklı sınıflara ayırmak amacıyla kullanılan gözetimli öğrenme yöntemlerinden birisidir. [0] Bu doğru yerleştirilirken asıl amaç doğrunun iki sınıfın noktaları için de maksimum uzaklıkta yer almasıdır.

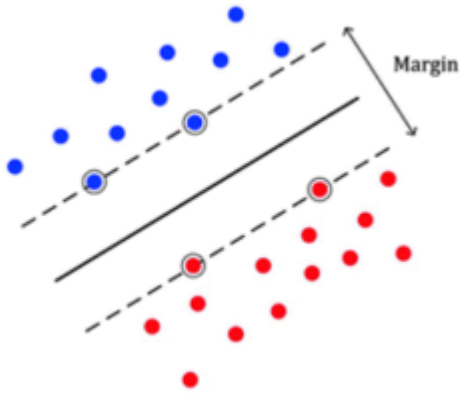

Fig. 9. SVM

Bu uzaklık ne kadar fazlaysa sınıflar da birbirinden o kadar iyi ayrılır. [0]

Formülü inceleyecek olursak:

$$\hat{y} = \begin{cases} 0 \text{ if } \mathbf{w}^T \cdot \mathbf{x} + b < 0, \\ 1 \text{ if } \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \end{cases}$$

Fig. 10. Formül

W: ağırlık vektörü
X: girdi vektörü
b: sapma miktarı

Şekil 1'deki görselden örnek verecek olursak formüle bir girdi verildiğinde çıkan sonuç 0'dan küçükse kırmızı noktalara daha yakın olacak şekilde sınıflandırılacaktır. Eğer çıkan sonuç 0'dan büyükse mavi noktalara daha yakın olacak şekilde sınıflandırılacaktır.

B. SVM Modelinin Kullanımı

Verilerin eşit dağılmayıp bir tarafta toplanmaları durumunda model yeterli performansı göstermeyip ortaya kötü bir sonuç çıkartabilir. Böyle bir durumda standardizasyon metodu uygulanarak dağılımın normale yaklaştırılmaya çalışılması gerekir. Bunun için sklearn kütüphanesinden StandartScaler (Fig 8) fonksiyonu kullanılarak kod üzerinde bu sorun giderilmektedir. Daha sonra SVC modelini kullanarak daha önceden train ve test diye belirli bir oranda ayırdığımız veri setimizin train kısmını ve bunun etiketlerini modele veriyoruz ve fit fonksiyonu yardımıyla oluşan nesne ile veri uyumunu sağlıyoruz. Modelin başarısını artırmak için doğru iterasyon sayısıyla birlikte modeli uygulamak önemli. Bunu deneyerek bulabiliriz. Daha sonra predict fonksiyonu kullanarak test verisiyle tahmin işlemini gerçekleştiriyoruz. En son modelin doğru tahminde bulunup bulunmadığını sklearn kütüphanesindeki metrikleri kullanarak ölçüyoruz.

Özellikle bu modelde başarı oranını artıracak uygun bir iterasyon sayısı bulamadığımız için verisetini balanced hale getirip modeli tekrar eğitmeye çalıştık.

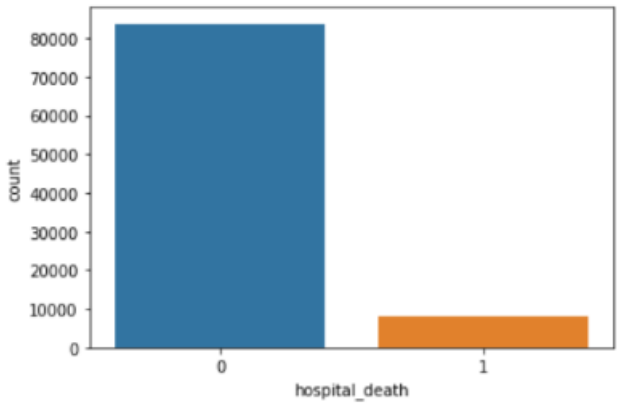

Fig. 11. imbalanced data

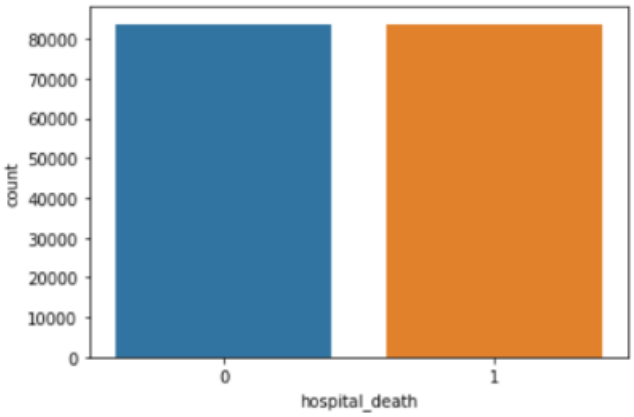

Fig. 12. balanced data

İstenilen sonuçlara bu şekilde de ulaşılamadığı için verisetindeki dağılımı incelemek adına PCA uygulamaya karar verdik. Boyut sayımızı 2'ye indirgeyip SVM modelinin verisetimizi ikiye ayırabilecek bir vektörünün olup olmayacağına bakmak istedik.

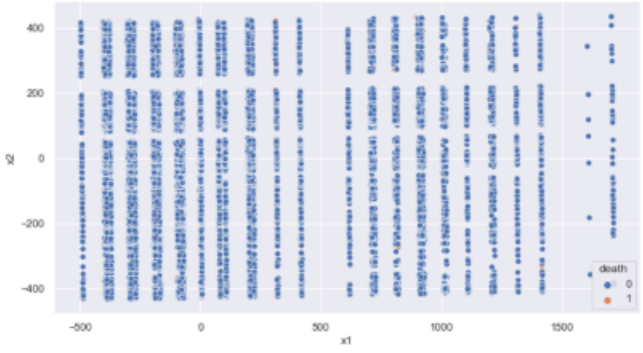

Fig. 13. SVM'in kodu

fig 11'de de görüldüğü üzere 1'lerin sayısı 0'larla eşit olmasına rağmen az miktarda ve bir vektörle bölünemeyecek kadar dağınık bir haldedir.

C. SVM Metrik değerleri

Precision: 0.4116
Recall: 0.4117
Accuracy: 0.4117
F1 Score: 0.4115

	precision	recall	f1-score	support
0	0.41	0.39	0.40	16796
1	0.41	0.43	0.42	16724
accuracy			0.41	33520
macro avg	0.41	0.41	0.41	33520
weighted avg	0.41	0.41	0.41	33520

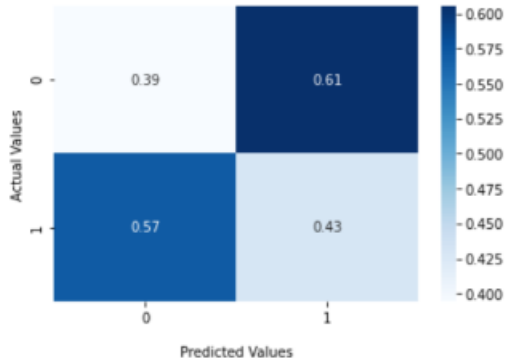

Fig. 14. SVM - Confusion Matrix çıktısı

IV. Lojistik Regresyon

A. Lojistik Regresyon Nedir?

Lojistik Regresyon, ikili sınıflandırma yöntemlerinde sıklıkla kullanılan bir sınıflandırma algoritmasıdır. En önemli özelliği, sonucun sadece iki farklı değer alabilmesi durumunda çalışmasıdır. Lojistik regresyonun doğru bir şekilde uygulanabilmesi için veri setinin doğrusal olarak ayrılabiliyor olması gerekmektedir. [0]

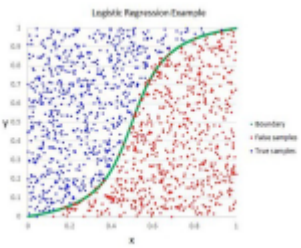

Fig. 15. Lojistik Regresyon

Lojistik regresyonla sınıflandırma yapabilmek için aktivasyon fonksiyonu olarak sigmoid fonksiyonunu kullandık. Sigmoid fonksiyonu Şekil 2'de de gösterildiği gibi "S" şeklinde bir eğridir ve verileri 0 ile 1 arasına sıkıştırmak için kullanılır. [0]

Sigmoid fonksiyonunun formülü:

$$g(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\theta^T x)}}$$

Fig. 16. Sigmoid Formülü

$$h(x) = \begin{cases} > 0.5, \text{ eğer } \theta^T x > 0 \rightarrow \text{Sınıfı 1 olarak tahmin eder.} \\ < 0.5, \text{ eğer } \theta^T x < 0 \rightarrow \text{Sınıfı 0 olarak tahmin eder.} \end{cases}$$

Fig. 17. Tahmin Formülü

$$\theta^T x > 0 => h(x) > 0.5$$

'tir ve sınıf 1 olarak tahmin edilir.

$$\theta^T x < 0 => h(x) > 0.5$$

'tir ve sınıf 0 olarak tahmin edilir.

B. Lojistik Regresyon'un Kullanımı

Lojistik regresyon kod üzerinde uygulanmasında, LojistikRegression fonksiyonu train ve label kümesi ile eğitildikten ve predict fonksiyonu ve test kümesiyle tahmin işlemi tamamlandıktan sonra eğer yeterli başarı oranı elde edilmezse max_iter parametresi değiştirilerek başarı oranı yükseltilmeye çalışılır. En son sklearn kütüphanesindeki metrikler kullanılarak başarı oranı değerlendirilir. [0]

C. Lojistik Regresyon Metrik değerleri

```
Precision: 0.7983
Recall: 0.7979
Accuracy: 0.7980
F1 Score: 0.7979
              precision    recall  f1-score   support

           0       0.79      0.82      0.80     16796
           1       0.81      0.78      0.79     16724

    accuracy                           0.80     33520
   macro avg       0.80      0.80      0.80     33520
weighted avg       0.80      0.80      0.80     33520
```

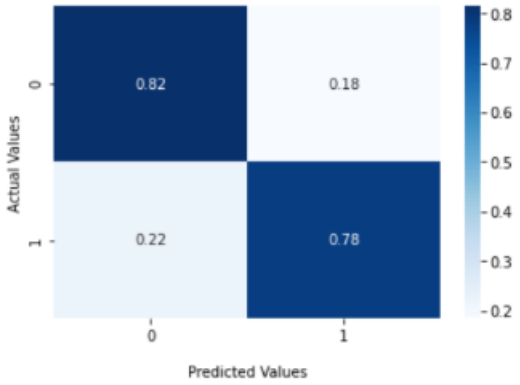

Fig. 18. Lojistik Regresyon - Confusion Matrix çıktısı

V. Stochastic Gradient Descent

A. SGD sınıflandırıcı Nedir?

SGD, her adımda rastgele bir veri üzerinde işlem yaparak global minimum değerine ulaşmayı amaçlayan optimizasyon algoritmalarından biridir. [0] SGD'nin algoritması:

$$\text{for } i \text{ in range } (m):$$
$$\theta_j = \theta_j - \alpha \left(\widehat{y}^i - y^i\right) x_j^i$$

Fig. 19. SGD

Her iterasyonda tek bir örneğin maliyet fonksiyonunun gradyanını buluruz [0]

B. SGD Sınıflandırıcısının Kullanımı

SGD sınıflandırıcısının kod üzerinde uygulanmasında, loss, penalty ve max_iter parametreleri başarı oranını artıracak şekilde denenir ve uygun değerler ile SGD modeli çalıştırılır. Model train ve label ile fit edilip daha sonra da predict fonksiyonu ve test kümesiyle tahmin işlemi gerçekleştikten sonra sklearn kütüphanesindeki metrikler kullanılarak sistemin başarısı değerlendirilir.

C. SGD - Metrik değerleri

```
Precision: 0.7928
Recall: 0.7923
Accuracy: 0.7923
F1 Score: 0.7922
              precision    recall  f1-score   support

           0       0.78      0.81      0.80     16796
           1       0.80      0.77      0.79     16724

    accuracy                           0.79     33520
   macro avg       0.79      0.79      0.79     33520
weighted avg       0.79      0.79      0.79     33520
```

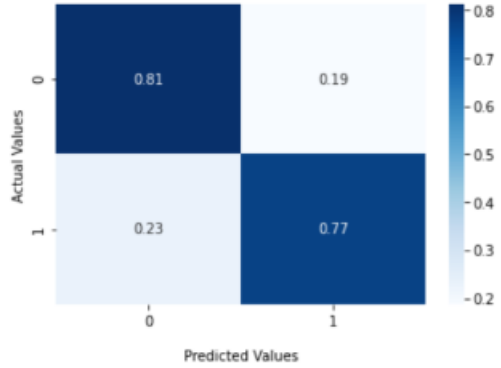

Fig. 20. SGD için Confusion Matrix çıktısı

Eğittiğimiz bütün modellerin sonuçlarının tek bi tabloda toplanmış şekli Fig 21'de gösterilmiştir.

	Recall	Precision	Accuracy	F1-Score
SVM	0.41	0.41	0.41	0.41
SGD	0.79	0.79	0.79	0.79
Logistic Regression	0.8	0.8	0.8	0.8

Fig. 21. Modellerin sonuçları

VI. GÖRSELLEŞTIRME

Riskli ve risksiz hastaların maksimum kalp atış hızı, maksimum solunum hızı, maksimum kan basıncı, vücut kitle indeksi ve maksimum glikoz seviyelerinin sütun grafiği şeklinde karşılaştırılması Fig 23'de gösterilmiştir.

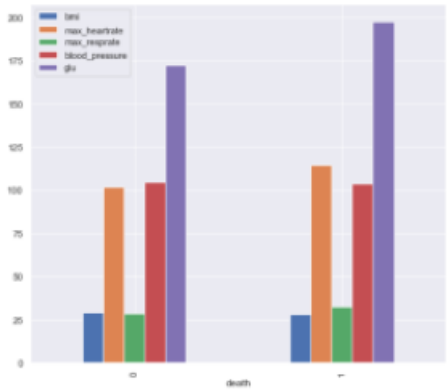

Fig. 22. Riskli ve Risksiz grupların genel dağılım grafiği

Yaş, vücut kitle indeksi ve sıcaklık apache skoru değerlerinin dağılımına göre riskli ve risksiz grupların karşılaştırılması Fig 25, Fig 26 ve Fig 27'da gösterilmiştir.

Fig. 23. yaş dağılımları

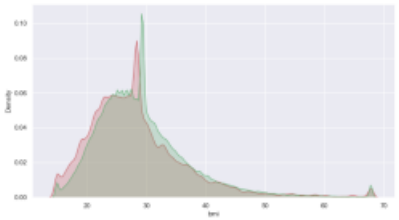

Fig. 24. vücut kitle indeksi dağılımları

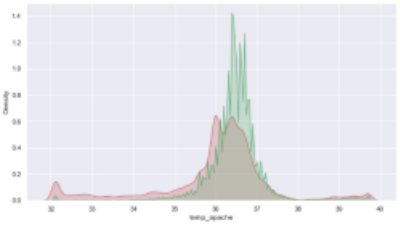

Fig. 25. sıcaklık apache skoru dağılımları

Hedef sütun ile korelasyon değerleri mutlakça 0.1'den yüksek olan özellikler Fig 29'deki gibi gösterilmiştir.

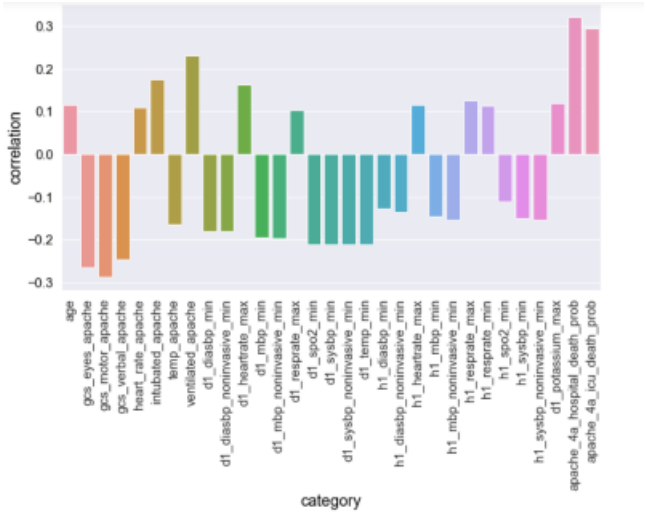

Fig. 26. Özelliklerin hedef sütun ile olan korelasyonlarının değerleri

VII. Conclusion

Bu çalışmada bir hastanın özelliklerine bakılarak o hasta için ölüm teşhisinin konulup konmayacağını tahminleyen modeller geliştirilmeye çalışılmıştır. Genel olarak diğer verisetlerine kıyasla çok daha fazla özelliği olan bir veriseti kullanılmıştır. Konu itibariyle ölüm gibi ciddi bir teşhisin çok fazla etkeninin olması aslında beklenen bir durumdur. Çalışmanın alanı sağlık olduğundan dolayı da diğer alanlara nazaran daha dikkatli olunması gerektiği düşünülerek modellerin bütün parametreleri dikkate alması tercih edilmiştir. Başarım metrikleri olarak daha ziyade recall ve precision dikkate alınmıştır. Recall metriği ölecek hastaların kaçına ölecek dediğimizi göstermektedir. Tedavi gerektiren hastaları kaçırmamak adına recall değeri oldukça önem arz etmektedir. Precision metriğiyse ölecek dediklerimizin kaçının öleceğini göstermektedir. Aslında ölüm durumu olmayan bir hastaya, öleceğini söylemek de psikolojik olarak riskli olduğundan bu değer de önem arz etmektedir. İki metrik değerine bakıldığında eğitilen modeller arasından en iyi sonucu Lojistik Regresyon modelinin verdiği görülmüştür.

References

[1] Akanksha$_{R}ai.(2021)Workingwithmissingdatainpandas.[Online].Available$:

N. Tamboli. (2021) All you need to know about different types of missing data values and how to handle it. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/

M. F. AKCA. (2020) Nedir bu destek vektör makineleri?). [Online]. Available: https://medium.com/deep-learning-turkiye/nedir-bu-destek-vekt%C3%B6r-makineleri-makine-%C3%B6%C4%9Frenmesi-serisi-2-94e576e4223e

E. ŞİRİN. (2017) Support vector machine (svm) ile sınıflandırma: Python Örnek uygulaması. [Online]. Available: https://www.veribilimiokulu.com/support-vector-machine-svm-ile-siniflandirma-python-ornek-uygulamasi/

E. Hatipoglu. (2018) Machine learning — classification — logistic regression — part 8. [Online]. Available: https://www.veribilimiokulu.com/veri-hazirliginin-vazgecilmezi-ozellik-olceklendirme/

M. F. AKCA. (2021) Lojistik regresyon nedir? nasıl Çalışır? [Online]. Available: https://mfakca.medium.com/lojistik-regresyon-nedir-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-4e1d2951c5c1/

S. M. Dereli. (2020) Veri hazırlığının vazgeçilmezi : Özellik Ölçeklendirme. [Online]. Available: https://www.veribilimiokulu.com/veri-hazirliginin-vazgecilmezi-ozellik-olceklendirme/

M. F. AKCA. (2020) Gradient descent nedir? [Online]. Available: https://medium.com/deep-learning-turkiye/gradient-descent-nedir-3ec6afcb9900

Rahul$_{R}oy.(2021)Ml|stochasticgradientdescent(sgd).[Online].Available$: