T.C.
KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

HEART FAILURE PREDICTION

PROJE ÖDEVİ

Hüseyin TAŞ: 18110131021

Muhsin Deniz: 18110131054

2021

Özet

Yapay zeka (AI), makinelerin deneyiminden öğrenmesini, yeni girdilere uyum sağlamasını ve insan benzeri görevleri gerçekleştirmesini mümkün kılar. Yapay zeka, büyük miktarda veriyi hızlı, yinelemeli işleme ve akıllı algoritmalarla birleştirerek çalışır ve yazılımın verilerdeki desenlerden veya özelliklerden otomatik olarak öğrenmesini sağlar. Yapay zeka, birçok teori, yöntem ve teknolojinin yanı sıra aşağıdaki ana alt alanları içeren geniş bir çalışma alanıdır.

1.Giriş

Kardiyovasküler hastalıklar (KVH), her yıl tahminen 17,9 milyon can alarak, dünya çapındaki tüm ölümlerin %31'ini oluşturan, küresel olarak 1 numaralı ölüm nedenidir. 5CVD ölümlerinden dördü kalp krizi ve felç nedeniyledir ve bu ölümlerin üçte biri 70 yaşın altındaki kişilerde erken meydana gelir. Kalp yetmezliği, CVD'lerin neden olduğu yaygın bir olaydır ve bu veri seti, olası bir kalp hastalığını tahmin etmek için kullanılabilecek 11 özellik içerir. Kardiyovasküler hastalığı olan veya yüksek kardiyovasküler risk altında olan kişiler (hipertansiyon, diyabet, hiperlipidemi veya halihazırda kurulmuş bir hastalık gibi bir veya daha fazla risk faktörünün varlığı nedeniyle), bir makine öğrenimi modelinin çok yardımcı olabileceği erken tespit ve yönetime ihtiyaç duyar.Veri setini açık veri kaynağı olan kaggle sitesinden indirdim. Bu veri seti heart failure prediction dataset (kalp yetmezliği) ile ilgilidir.Bu yüzden bu projemizde bu heart failure prediction datasetini kullanarak kalp yetmezliğini tahmin eden bir web uygulaması yapacağız.

2. Veri Seti Oluşturma ve Düzenleme

Veri setini Kaggle sitesinden indirdikten sonra veri setini import ediyorum. Ve veri setine genel bir bakış atıyorum.

```
# 1. Load and Check data

data = pd.read_csv("dataset/heart.csv")
print(data.head())
# veri hakkında istatistikler
print(data.describe().T)
print(data.shape)
```

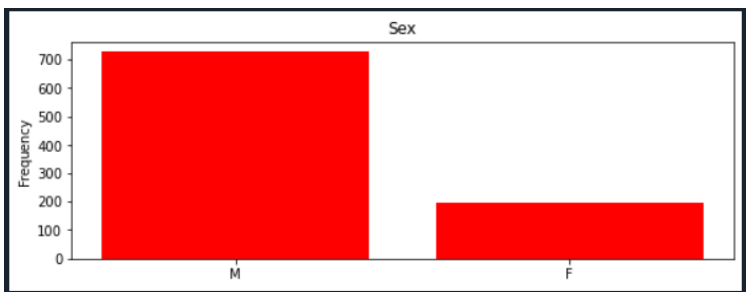
```
   Age Sex ChestPainType  ...  Oldpeak  ST_Slope  HeartDisease
0   40   M           ATA  ...      0.0        Up             0
1   49   F           NAP  ...      1.0      Flat             1
2   37   M           ATA  ...      0.0        Up             0
3   48   F           ASY  ...      1.5      Flat             1
4   54   M           NAP  ...      0.0        Up             0

[5 rows x 12 columns]
              count        mean         std   min     25%    50%    75%    max
Age           918.0   53.510893    9.432617  28.0   47.00   54.0   60.0   77.0
RestingBP     918.0  132.396514   18.514154   0.0  120.00  130.0  140.0  200.0
Cholesterol   918.0  198.799564  109.384145   0.0  173.25  223.0  267.0  603.0
FastingBS     918.0    0.233115    0.423046   0.0    0.00    0.0    0.0    1.0
MaxHR         918.0  136.809368   25.460334  60.0  120.00  138.0  156.0  202.0
Oldpeak       918.0    0.887364    1.066570  -2.6    0.00    0.6    1.5    6.2
HeartDisease  918.0    0.553377    0.497414   0.0    0.00    1.0    1.0    1.0
(918, 12)
```

Veri seti hakkında bilgiler elde ettim. Bu aşamada veri setindeki özniteliklerin açıklayacağım.

- Age -> Hastanın yaşı
- Sex -> Hastanın cinsiyeti
- ChestPainType -> Göğüs ağrısı tipi [TA: Tipik Angina, ATA: Atipik Angina, NAP: Anjinal olmayan ağrı, ASY: Asemptomatik]
- RestingBP -> Dinlenme kan basıncı [mm Hg]
- Cholesterol -> Serum kolesterölü [mm/dl]
- FastingBS -> Açlık kan şekeri [1: if FastingBS > 120 mg/dl, 0: otherwise]
- RestingECG -> Dinlenme elektrokardiyogram sonuçları [Normal: Normal, ST: ST-T dalga anormalliği olan T dalgası inversiyonları ve/veya ST elevasyonu veya depresyonu > 0.05 mV), LVH: Estes kriterlerine göre olası veya kesin sol ventrikül hipertrofisini gösteriyor]
- MaxHR -> Ulaşılan maksimum kal atış hızı [60 ile 202 arasında sayısal değer]
- ExerciseAngina -> egzersize bağlı angina [Y: Evet, N: Hayır]
- Oldpeak -> oldpeak = ST [Depresyonda ölçülen sayısal değer]
- ST_Slope -> zirve egzersiz ST segmentinin eğimi [Up: upsloping(eğimli), Flat: flat(düz), Down: dowmsloping(aşağı eğimli)]
- HeartDisease -> output class [1: heart disease, 0: Normal]

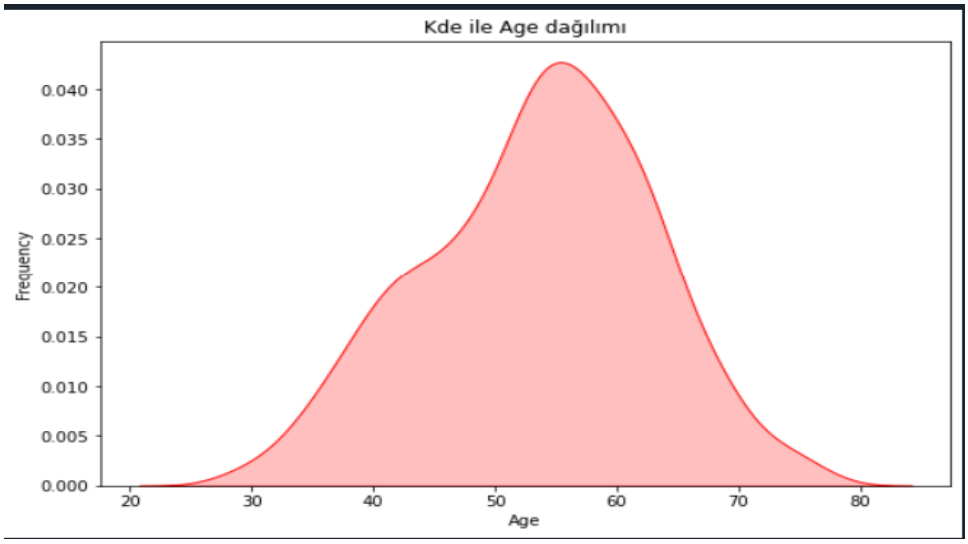
Veri setindeki kategorik ve sayısal değişkenlerin dağılımını görmek için veri görselleştirme işlemi yaptım. Veri görselleştirme için seaborn ve matplotlib kütüphanelerini kullandım. Kategorik değişkenlerin dağılımını görmek için bar plot kullandım.

Örneğin:



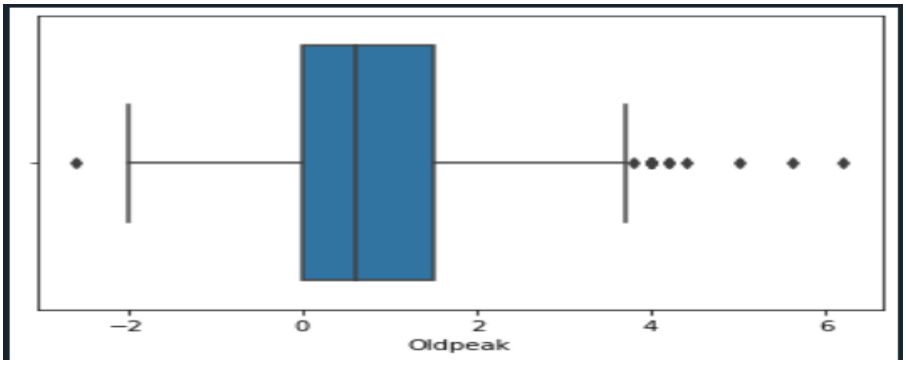
Sayısal değişkenlerin dağılımını görmek için kde plot kullandım.

Örneğin:



Veri setindeki aykırı değerlerin tespitini (outlier detection) hem box plot ile veri görselleştirme kullanarak hem de eşik değerleri belirleyerek aykırı değer tespitini gerçekleştirdim.

```
# 5. Outlier Detection

# plot ile aykırı değer tespiti
list_data = ["Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak"]
for i in list_data:
    sns.boxplot(x=data[i])
    plt.show()


# Aykırı değerleri alt ve üst eşik değerlere baskılama
def find_quantile(df, variable):
    IQR = df[variable].quantile(0.75) - df[variable].quantile(0.25)

    alt_sinir = df[variable].quantile(0.25) - (IQR * 1.5)
    ust_sinir = df[variable].quantile(0.75) + (IQR * 1.5)
    return alt_sinir, ust_sinir

age_alt_sinir, age_ust_sinir = find_quantile(data, "Age")
resting_alt_sinir, resting_ust_sinir = find_quantile(data, "RestingBP")
chol_alt_sinir, chol_ust_sinir = find_quantile(data, "Cholesterol")
max_alt_sinir, max_ust_sinir = find_quantile(data, "MaxHR")
peak_alt_sinir, peak_ust_sinir = find_quantile(data, "Oldpeak")


def count_outlier(df, variable, alt_sinir, ust_sinir):
    x = df[df[variable] < alt_sinir][variable].size
    y = df[df[variable] > ust_sinir][variable].size
    print(variable, "Outlier sayısı: ", x + y)


count_outlier(data, "Age", age_alt_sinir, age_ust_sinir)
count_outlier(data, "RestingBP", resting_alt_sinir, resting_ust_sinir)
count_outlier(data, "Cholesterol", chol_alt_sinir, chol_ust_sinir)
count_outlier(data, "MaxHR", max_alt_sinir, max_ust_sinir)
count_outlier(data, "Oldpeak", peak_alt_sinir, peak_ust_sinir)
```



- Age outlier sayısı -> 0
- RestingBP Outlier sayısı: 28
- Cholesterol Outlier sayısı: 183
- MaxHR Outlier sayısı: 2
- Oldpeak Outlier sayısı: 16

Aykırı değerleri tespit ettikten sonra istersek bu aykırı değerleri silebiliriz veya bu aykırı değerleri alt eşik değere veya üst eşik değere baskılayabilirim. Ama ben bu projemde aykırı değerlere bir işlem yapmayacağım. Bu projede aykırı değerleri göz ardı edeceğim.

Veri setinde eksik veri olup olmadığını kontrol ediyorum. Bizim veri setimizde eksik veri yok.

Bu aşamada ise kategorik verileri modelimize uygun hale getirmek için sayısal hale çevirdim.

```
# Encoding
# kategorik verileri sayısal hale çevirme
data["Sex"] = data.Sex.map({"M": 0, "F": 1})
data["ChestPainType"] = data.ChestPainType.map({"TA": 0, "ATA": 1, "NAP": 2, "ASY": 3})
data["RestingECG"] = data.RestingECG.map({"Normal": 0, "ST": 1, "LVH": 2})
data["ExerciseAngina"] = data.ExerciseAngina.map({"Y": 1, "N": 0})
data["ST_Slope"] = data.ST_Slope.map({"Up": 1, "Flat": 0, "Down": 2})
```

Veri setim artık modele sokmak için uygun hale getirdim. Artık veri setimi eğitebilirim.Veri setini train ve test split olarak ayırdım. Burada %30 test, %70 train seti olarak ayırdım. Ve modelimizin daha hızlı çalışması ve hesaplama maliyetini azaltmak için X_train ve X_test StandartScaler kullanarak standardizasyon işlemi gerçekleştirdim.

```
# %%

# train - test split

X = data.drop("HeartDisease", axis=1)
y = data["HeartDisease"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.30,
                                                    random_state=42)


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

3. Ağ Eğitme

Veri setini eğitmek için KNN ve yapay sinir ağları modellerini uyguladım.

3.1 KNN ile Model Oluşturma

Algoritma, sınıfları belli olan bir örnek kümesindeki verilerden yarrarlanarak kullanılmaktadır. Önek veri setine katılacak olan yeni bir verinin, mevcut verilere göre uzaklığı hesaplanıp, k sayıda yakın komşuluğuna bakılırKNN; en basit ve gürültülü eğitim verilerine karşı dirençli olduğu için en popüler ML algoritmalarından biridir. Fakat bunun yanında dezavantajı; uzaklık hesabı yaparken bütün durumları hesapladığından, büyük veriler için kullanıldığında çok sayıda bellek alanına gereksinim duyar.

```
# KNN

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn_model = knn.fit(X_train, y_train)

y_pred = knn_model.predict(X_test)
print("Accuracy Score: ", accuracy_score(y_test, y_pred))


# Stratified K-Fold cross validation

skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Hyperparametre tuning with GridSearchCV
knn_params = {"n_neighbors": np.arange(1, 50)}
knn = KNeighborsClassifier()
knn_cv_model = GridSearchCV(knn, knn_params, cv=skf, scoring="accuracy")
knn_cv_model.fit(X_train, y_train)

print("En iyi skor: {}".format(knn_cv_model.best_score_))
print("En iyi K değeri {}".format(knn_cv_model.best_params_))

# En iyi parametre ile tuned edilmiş modeli kurma
knn = KNeighborsClassifier(n_neighbors=31)
knn_tuned = knn.fit(X_train, y_train)

y_pred = knn_tuned.predict(X_test)
y_pred_train = knn_tuned.predict(X_train)
print("Test Accuracy Score: ", accuracy_score(y_test, y_pred))
print("Train Accuracy Score: ", accuracy_score(y_train, y_pred_train))

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))




score = cross_val_score(knn_tuned, X_train, y_train, cv=skf, scoring="accuracy")
print("Train score degerleri: ", score)
print("Train  Mean Score:", score.mean())

score = cross_val_score(knn_tuned, X_test, y_test, cv=skf, scoring="accuracy")
print("Test score degerleri: ", score)
print("Test Mean Score:", score.mean())

pickle.dump(knn_tuned, open("knn_model.pkl", "wb"))
```

Burada ilk başta KNN algoritmasının hiçbir parametresine müdahale etmeden modeli eğitiyoruz. Ve burada ilk ilkel test hatamızı elde ediyoruz. Bu durumda KNN'nin k değeri otomatik olarak 5' tir.

Daha sonra en optimum k değerini bulmak için GridSearchCV kullaranak k değerini 1 ile 50 arasında değerleri tek tek deneyerek en optimum k değerini elde ediyoruz. Bizim modelimizde en optimum k değerini 31 olarak bulduk. Bu bulduğumuz k değeri ile doğrulanmış modelimizi yeniden oluşturuyoruz ve sonuçlarımızı elde ediyoruz.

```
En iyi skor: 0.8660576923076924
En iyi K değeri {'n_neighbors': 31}
Test Accuracy Score:  0.8623188405797102
Train Accuracy Score:  0.8644859813084113
[[102  10]
 [ 28 136]]
              precision    recall  f1-score   support

         0.0       0.78      0.91      0.84       112
         1.0       0.93      0.83      0.88       164

    accuracy                           0.86       276
   macro avg       0.86      0.87      0.86       276
weighted avg       0.87      0.86      0.86       276
```

Daha sonra overfitting olayını engellemek için Stratified K-Fold cross validation (SKF cross validation) işlemini kullanıyoruz. Bunun amacı veri setini train ve test seti olarak ikiye ayırırken veri setini rastgele ayırmamış olabiliriz. Yani veri setini bölerken sadece kadınları almış olabilir veya sadece belli bir özelliği almış olabilir. Bu yüzden cross validation işlemini kullanacağız. SKF cross validation modelin yüksek performansının rastgele olup olmadığını görmemizi sağlayacaktır.

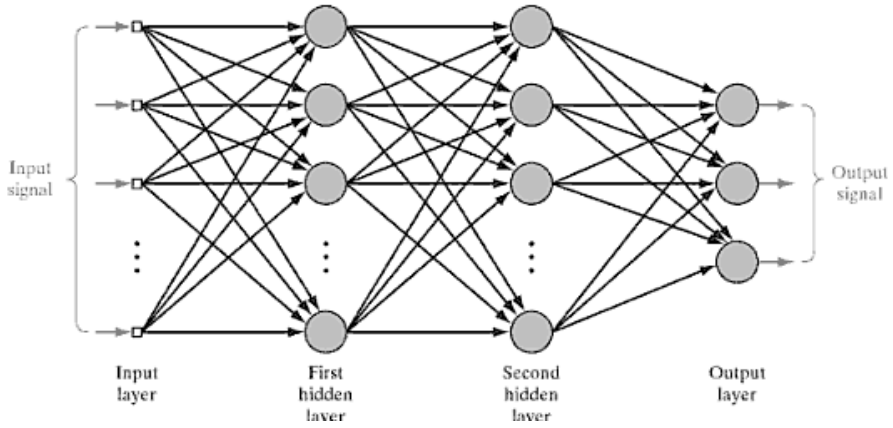
SKF cross validation hem modelimizin kalitesini görmemizi hem de overfitting problemi ile karşı karşıya olup olmadığımızı sağlayacaktır. Bu işlemi projemde uyguladığımda;

```
Train score degerleri:  [0.87692308 0.84615385 0.890625   0.859375   0.90625    0.890625
 0.90625    0.875      0.8125     0.796875  ]
Train  Mean Score: 0.8660576923076924
Train Std : 0.03568117521597207
Test score degerleri:  [0.89285714 0.92857143 0.96428571 0.82142857 0.82142857 0.85714286
 0.77777778 0.88888889 0.88888889 0.85185185]
Test Mean Score: 0.8693121693121693
Test Std : 0.052220372749932895
```

SKF cross validation uyguladıktan sonra bizim modelizin başarı oranının rastgele olmadığını ve modelimizin kalitesini görmüş oluyoruz.

3.2 Yapay Sinir Ağları ile Model Oluşturma

Yapay Sinir Ağı (YSA), yapay nöronlar adı verilen bağlı birimlerin koleksiyonuna dayalı yapay beyinler oluşturmak için biyolojik sinir ağlarından esinlenen bir bilgisayar sistemidir. Bilgiyi insanlar olarak analiz etmek ve işlemek için tasarlanmıştır. Yapay Sinir Ağı, daha fazla veri mevcut olduğundan daha iyi sonuçlar üretmek için kendi kendine öğrenme yeteneklerine sahiptir.



Oluşturduğumuz ver iseti kalp yetmezliği ile ilgilidir. Bu veri setimizde 11 adet özellik (input) ve 1 tane çıkış vardır. Bu veride 11 özellik 918 kişiden alınıyor.

İlk başta en optimum parametreleri bulmak için bir yapay sinir ağı fonksiyonu ve GridSearchCV kullanarak en optimum parametreleri buldum.

Biz burada kendimiz elle birkaç tane parametre belirliyoruz. Ve bu parametreler içinde en iyi kombinasyonu GridSearchCV ile buluyoruz.

```
layers = [[16, 16], [32, 16, 32], [32, 16, 16]]

params = {"layers": layers,
          "batch_size": [32, 64],
          "epochs": [100, 200, 300]}
```

```python
# YSA
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras import callbacks
from tensorflow.keras.optimizers import Adam
from keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.activations import relu, sigmoid


def build_model(layers):
    model = Sequential()
    for i, nodes in enumerate(layers):
        if i == 0:
            # Input layer - first hidden layer
            model.add(Dense(16, input_dim=11, activation="relu"))
            model.add(Dropout(0.25))
        else:
            model.add(Dense(nodes))
            model.add(Activation('relu'))
            model.add(Dropout(0.25))
    model.add(Dense(1, kernel_initializer="uniform", activation="sigmoid"))
    model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
    return model


earlystopping = callbacks.EarlyStopping(monitor='val_loss',
                                        mode='min',
                                        verbose=1,
                                        patience=20)
model = KerasClassifier(build_fn=build_model,
                        callbacks=[earlystopping],
                        validation_data=(X_test, y_test))


skf = StratifiedKFold(n_splits=10, random_state=42, shuffle=True)

layers = [[16, 16], [32, 16, 32], [32, 16, 16]]

params = {"layers": layers,
          "batch_size": [32, 64],
          "epochs": [100, 200, 300]}

grid_cv = GridSearchCV(model, param_grid=params, cv=skf, scoring='accuracy')
grid_cv.fit(X_train, y_train,
            validation_data=(X_test, y_test),
            callbacks=[earlystopping])
```

Bu kod ile en iyi parametreyi buluyorum. Bu işlem biraz uzun sürüyor. Bu yüzden GridSearchCV yerine RandomizeSearchCV veya manuel olarak parametreleri elle girerek denenebilir.

```python
print("En iyi parametreler: {}".format(grid_cv.best_params_))
print("En iyi Skor: {}".format(grid_cv.best_score_))

"""
En iyi parametreler: {'batch_size': 64, 'epochs': 100, 'layers': [32, 16, 16]}
En iyi Skor: 0.867548076923077
"""
```

Girdiğim parametreler içerisindeki en iyi parametre kombinasyonu ile yapay sinir ağı modelimiz oluşturuyorum.

```python
# En iyi parametreler ile tekrardan YSA eğitme
def create_model():
    ann_model = Sequential()
    # Input
    ann_model.add(Dense(32, activation='relu', input_dim=X_train.shape[1]))
    ann_model.add(Dropout(0.25))

    ann_model.add(Dense(16, activation='relu', kernel_initializer="uniform"))
    ann_model.add(Dropout(0.25))

    ann_model.add(Dense(16, activation='relu', kernel_initializer="uniform"))
    ann_model.add(Dropout(0.25))

    # Dropout
    ann_model.add(Dense(1, activation='sigmoid', kernel_initializer="uniform"))
    optimizer = Adam(learning_rate=0.001)
    ann_model.compile(optimizer=optimizer, loss="binary_crossentropy", metrics=["accuracy"])
    return ann_model


model = create_model()

history = model.fit(X_train, y_train,
                    batch_size=64, epochs=100,
                    validation_data=(X_test, y_test),
                    callbacks=[earlystopping])
```

Modelimi oluşturduktan sonra modelimin başarı oranını bakıyorum. Burada threshold değeri belirliyoruz. Threshold değeri olarak 0.5 ve 0.4 değerlerini belirleyip test doğruluğuna bakıyorum.

- Threshold 0.5 için;

```python
# load model
from keras.models import load_model

model = load_model("kerasmodel.h5")

# Model Performance

y_pred = model.predict(X_test)
prediction_label = [np.argmax(i) for i in y_pred]
binary_prediction = []

for i in y_pred:
    if i > 0.5:
        binary_prediction.append(1)
    else:
        binary_prediction.append(0)

print(classification_report(y_test, binary_prediction))
print(confusion_matrix(y_test, binary_prediction))
print("Model Score: ", accuracy_score(y_test, binary_prediction))

"""
              precision    recall  f1-score   support

         0.0       0.82      0.90      0.86       112
         1.0       0.93      0.87      0.90       164

    accuracy                           0.88       276
   macro avg       0.87      0.88      0.88       276
weighted avg       0.88      0.88      0.88       276

[[101  11]
 [ 22 142]]
Model Score:  0.8804347826086957
"""
```

- Threshold 0.4 için;

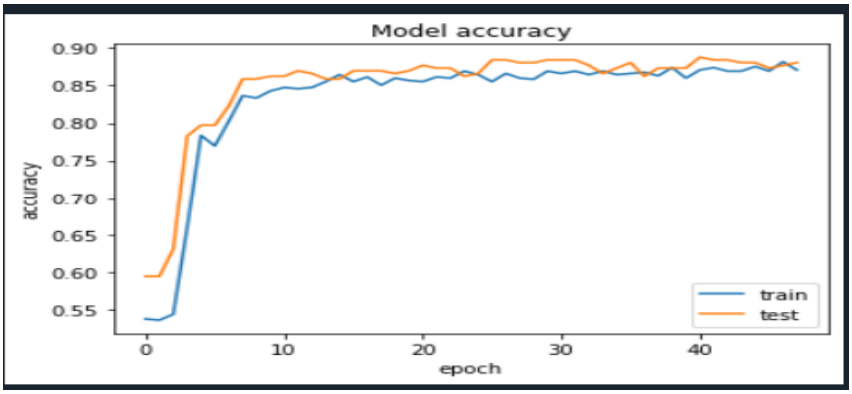
```
# load model
from keras.models import load_model

model = load_model("kerasmodel.h5")

# Model Performance

y_pred = model.predict(X_test)
prediction_label = [np.argmax(i) for i in y_pred]
binary_prediction = []

for i in y_pred:
    if i > 0.4:
        binary_prediction.append(1)
    else:
        binary_prediction.append(0)

print(classification_report(y_test, binary_prediction))
print(confusion_matrix(y_test, binary_prediction))
print("Model Score: ", accuracy_score(y_test, binary_prediction))

"""
              precision    recall  f1-score   support

         0.0       0.86      0.90      0.88       112
         1.0       0.93      0.90      0.91       164

    accuracy                           0.90       276
   macro avg       0.89      0.90      0.90       276
weighted avg       0.90      0.90      0.90       276

[[101  11]
 [ 17 147]]
Model Score:  0.8985507246376812
"""
```

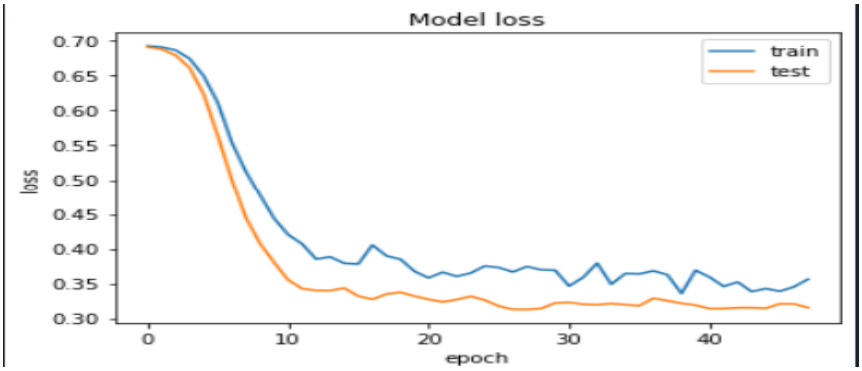
Elde ettiğimiz değerleri threshold değerleri ile karşılaştırdık. Elde ettiğimiz değerlerin threshold verileri ile karşılaştırdığımızda en uygun threshold değerini bulduk. Bizim kodumuzda en uygun threshold değeri 0.4 thresholdu uygun olmuştur.

Başarı oranımın rastgele olup olmadığını gözlemlemek için SKF cross validation işlemi kullanıyorum.

```
earlystopping = callbacks.EarlyStopping(monitor='val_loss',
                                        mode='min',
                                        verbose=1,
                                        patience=20)
keras_clf = KerasClassifier(create_model,
                            validation_data=(X_test, y_test),
                            epochs=100, batch_size=64,
                            callbacks=[earlystopping])

# Stratified K-Fold cross validation
skf = StratifiedKFold(n_splits=10, random_state=42, shuffle=True)

accuracies = cross_val_score(estimator=keras_clf,
                             X=X_train, y=y_train, cv=skf
                             , scoring="accuracy")
mean = accuracies.mean()
variance = accuracies.std()
print("Accuracy mean: " + str(mean))
print("Accuracy variancce: " + str(variance))
print(accuracies)

"""
Accuracy mean: 0.8519951923076924
Accuracy variancce: 0.027425145653035957
[0.86153846 0.86153846 0.875      0.84375    0.828125   0.875
 0.875      0.859375   0.78125    0.859375  ]
"""
```

Cross validation işlemi sonnrası modelimizin kalitesinin ve baarı oranının rastgele olmadığını görmüş oluyoruz.



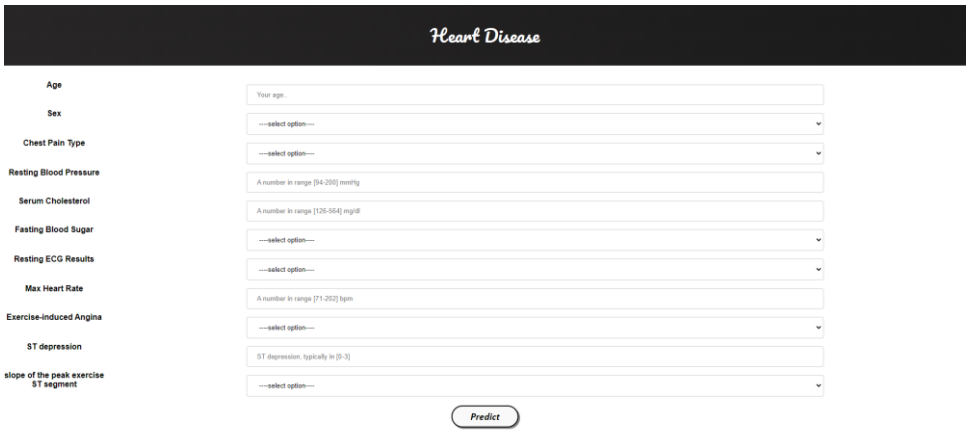


5. Modeli Web Sitesine Entegre Etme

HTML ve CSS kullanarak bir ara yüz oluşturduk daha sonra FLASK kullanarak modeli web sitesine entegrasyon işlemini gerçekleştirdim.

```
model = pickle.load(open("knn_model.pkl", "rb"))
app = Flask(__name__)


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


@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        age = int(request.form['Age'])
        sex = request.form.get('Sex')
        cp = request.form.get('ChestPainType')
        trestbps = int(request.form['RestingBP'])
        chol = int(request.form['Cholesterol'])
        fbs = request.form.get('FastingBS')
        restecg = request.form.get('RestingECG')
        maxheart = int(request.form['MaxHR'])
        exang = request.form.get('ExerciseAngina')
        oldpeak = int(request.form['Oldpeak'])
        slope = request.form.get('ST_Slope')
        data = np.array([[age, sex, cp, trestbps, chol, fbs, restecg, maxheart, exang, oldpeak, slope]])
        my_prediction = model.predict(data)
        if my_prediction > 0.5:
            my_prediction = 1
        else:
            my_prediction = 0
        return render_template('result.html', prediction=my_prediction)


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

- Web sitesinin görüntüleri;



5. Sonuç

Heart failure data seti seçtik. KNN ve yapay sinir ağları algoritmalarını kullanarak model oluşturduk. Model için en uygun hiperparametre değerlerini bulduk. Daha sonra modelimizi eğittikten sonra onun teste sokarak modelin başarı oranını ve performansını gözlemledik. En uygun threshold grafiğini belirledik ve sonucu belirledik. Daha sonra modelleri model.save ile kaydettiken sonra bunları Flask, HTML, CSS kullanarak bir web sitesine entegre ettik. Görüntüleri aşağıdadır.

Kaynaklar

[1] https://towardsdatascience.com/gridsearchcv-for-beginners-db48a90114ee

[2] https://www.analyticsvidhya.com/blog/2021/05/tuning-the-hyperparameters-and-layers-of-neural-network-deep-learning/

[3] https://www.kaggle.com/fedesoriano/heart-failure-prediction

[4] https://medium.com/@gulcanogundur/model-se%C3%A7imi-k-fold-cross-validation-4635b61f143c

[5] https://www.kaggle.com/satishgunjal/tutorial-k-fold-cross-validation#Inner-Working-of-Cross-Validation-

[6] https://seaborn.pydata.org/examples/index.html

[7] https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5