

لهم اجعلني
من عبادك
وأنت أنت
الحبيب الظاهر

تشخیص آریتمی قلبی با استفاده از الگوریتم های یادگیری ماشین

نگین فروزان

استاد راهنمای

۱۴۰۲/۱۱/۲۸

دانشجو

جناب آقای دکتر یزدی زاده

تاریخ ارائه

فهرست مطالب



بررسی و پردازش
مجموعه داده



مقدمه
پژوهش



نتیجه‌گیری
پیش‌بینی



روش‌شناسی



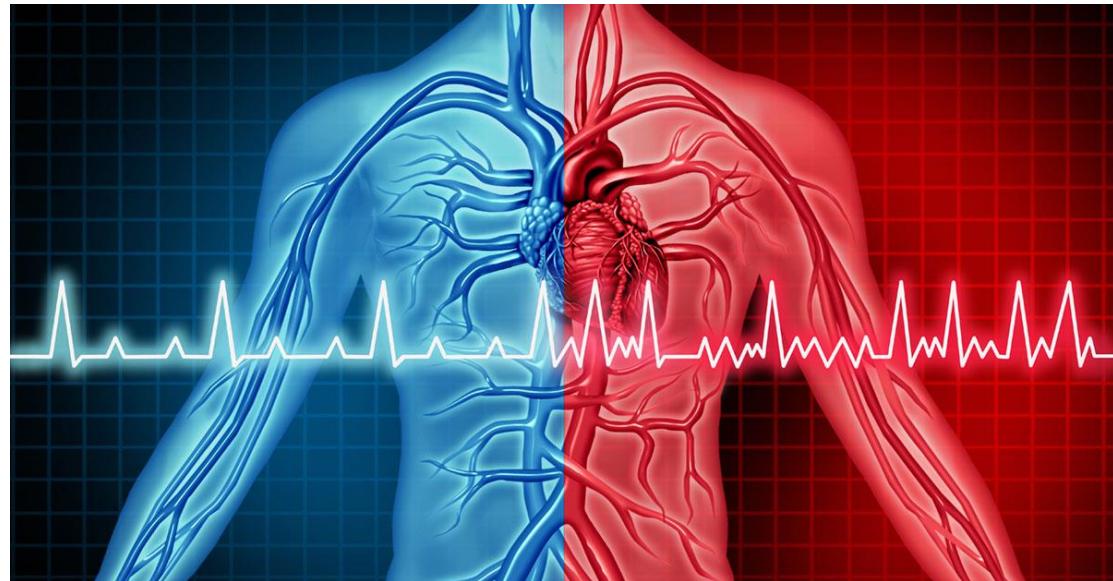
مقـدـمـه پـژـوهـش





بیماری‌های قلبی-عروقی از سال ۱۹۹۹ به عنوان علت اصلی مرگ شناخته شده‌اند، در حالی که ۹۰٪ از چنین بیماری‌هایی قابل پیشگیری هستند.

یکی از متداول‌ترین بیماری‌های قلبی-عروقی آریتمی قلبی است. آریتمی نوعی بیماری قلبی



است که در آن ضربان قلب از الگوی نرمال خود منحرف شده و یک الگوی نامنظم به خود می‌گیرد، به عبارتی نوعی بی‌نظمی در "سرعت" یا "ریتم" ضربان قلب است.

مقدمه پژوهش



بررسی و پردازش
مجموعه داده



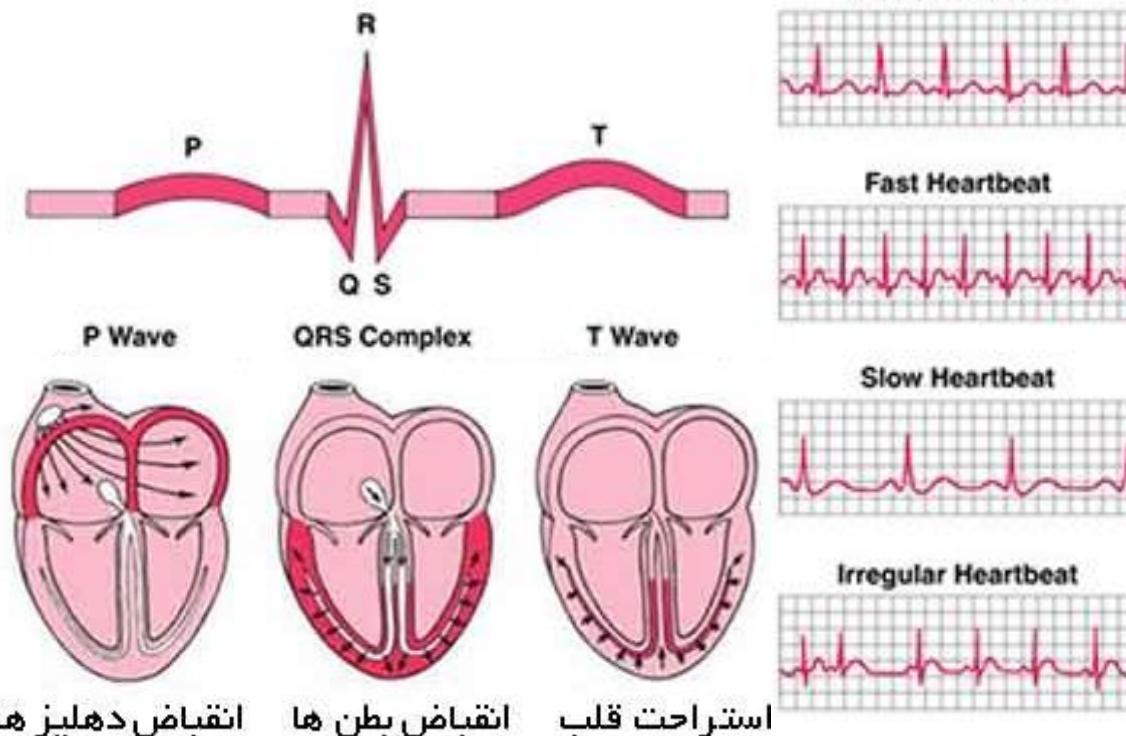
روش شناسی



نتیجه گیری



شناسایی و طبقه بندی دقیق آریتمی با استفاده از سیگنال های الکتروکاردیوگرام انجام می شود که با قرار دادن ۱۰ نقطه از بدن ساخته می شود و فعالیت الکتریکی قلب را ثبت می کند.



یک الکتروکاردیوگرام نرمال انحرافاتی به سمت بالا و پایین دارد که به ترتیب با حروف P، Q، R، S، T نشان داده می شوند.

هرگونه تغییر از حالت نرمال در الکتروکاردیوگرام، می تواند نشان دهنده اختلال قلبی باشد.

مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



تشخیص آریتمی با روش های مبتنی بر هوش مصنوعی

- ✓ نظارت بر سیگنال الکتروکاردیوگرام از راه دور
- ✓ نظارت به طور مداوم
- ✓ نظارت در بازه های زمانی طولانی مدت
- ✓ کاهش وابستگی به تفسیر انسانی
- ✓ شناسایی سریع الگوهای نامطلوب ریتم قلب
- ✓ مدیریت پیچیدگی داده های الکتروکاردیوگرام
- ✓ تشخیص سریع تر و درمان به موقع
- ✓ کاهش خطا در تشخیص آریتمی های پراکنده

تشخیص آریتمی با روش های سنتی

- اشتباه در تشخیص یا تأخیر در درمان
- اهمیت میزان مهارت تفسیر دکتر متخصص
- عدم ثبت دقیق تغییرات به طور کامل
- عدم ثبت آریتمی های پراکنده یا متناوب در ضبط های کوتاه مدت الکتروکاردیوگرام

مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



مراحل انجام پروژه

۵

۴

۳

۲

۱



مرحله اول

آماده سازی
سیگنال اولیه

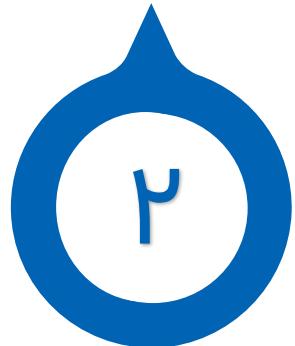


مرحله سوم

مقایسه و ارزیابی
الگوریتم ها

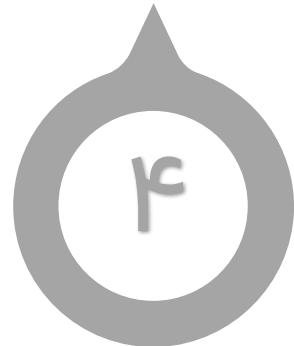


مرحله چهارم



انتخاب دیتاست

پیاده سازی الگوریتم های
یادگیری ماشین



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



مراحل انجام پروژه

۵

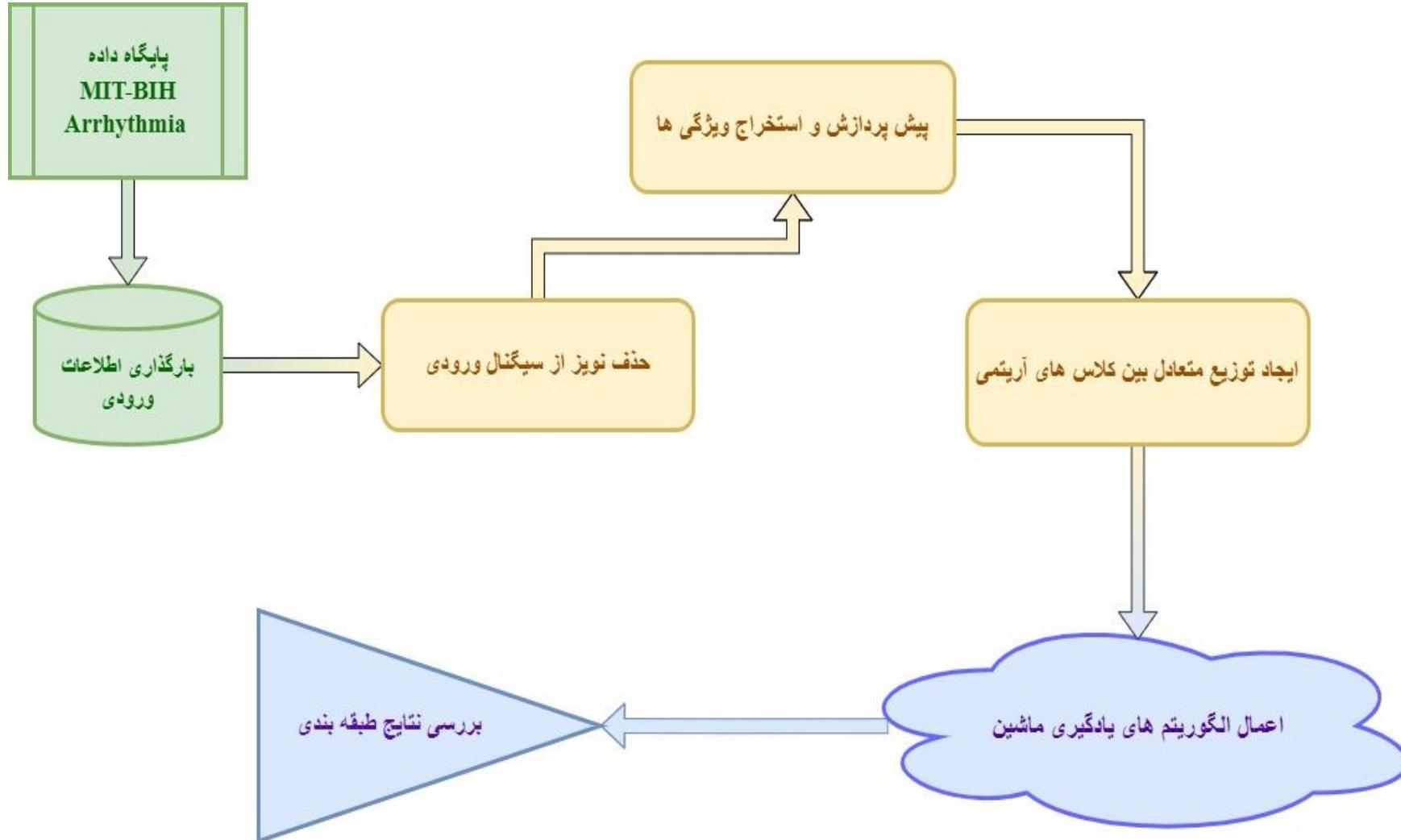
۴

۳

۲

۱

۹



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



بررسی و پردازش مجموعه داده





پایگاه داده MIT-BIH Arrhythmia متشکل از پنج کلاس است که مشخص کننده نوع آریتمی هستند

این کلاس ها عبارتند از:

مقدمه پژوهش



بررسی و پردازش
مجموعه داده



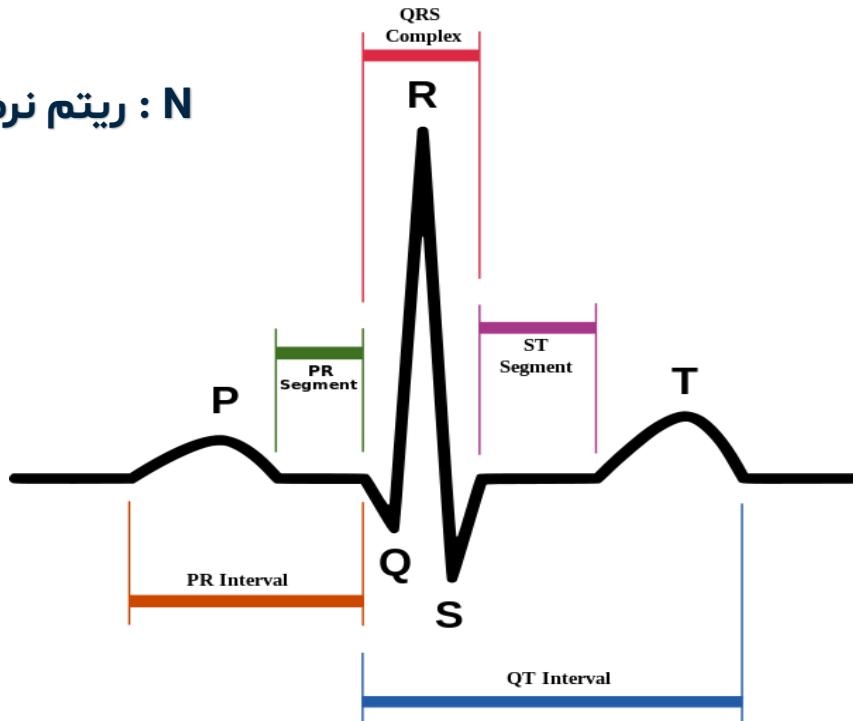
روش شناسی



نتیجه گیری



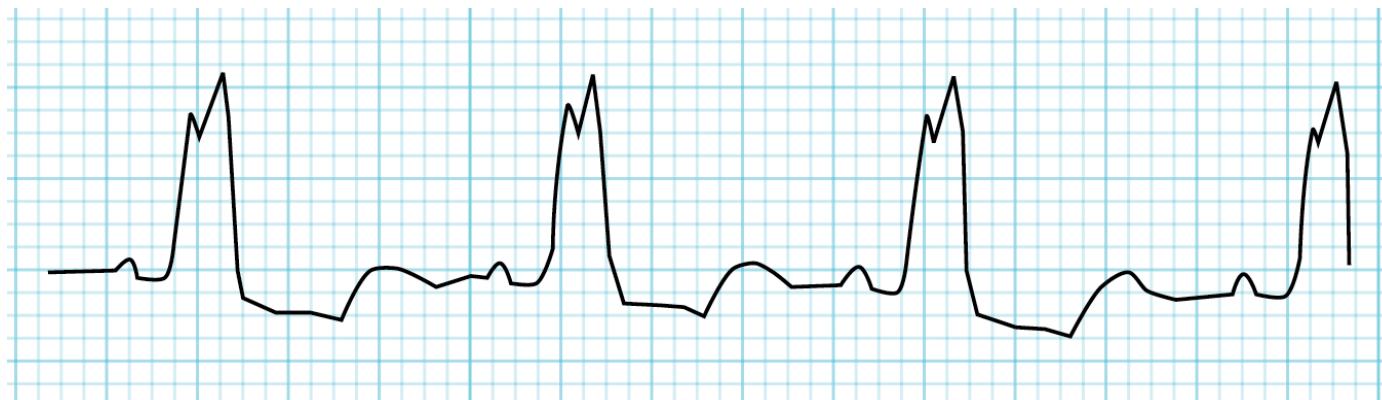
N : ریتم نرمال قلب



۳

R : بلوک شاخه ای راست
Right Bundle Branch Block(RBBB)

در حالت نرمال دو بطن با هم دیگر منقبض می شوند و یک موج R ثبت می کنند اما در بلوک شاخه ای بطن سالم زود تر منقبض می شود و یک موج R ثبت می کند. اواسط دپلاریزاسیون بطن سالم، بطن دیگر نیز جریان الکتریکی دریافت می کند و یک موج R دیگر ثبت می شود



L : بلوک شاخه ای چپ
Left Bundle Branch Block(LBBB)

۲

مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



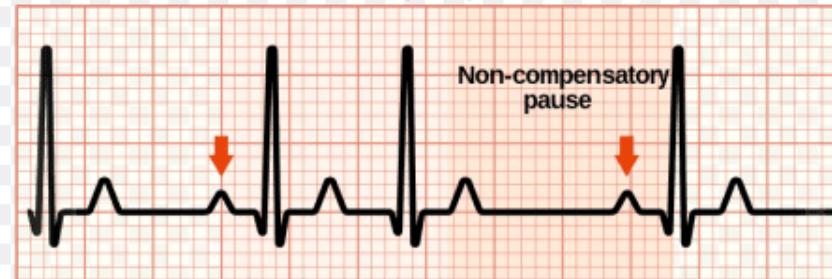
نتیجه گیری





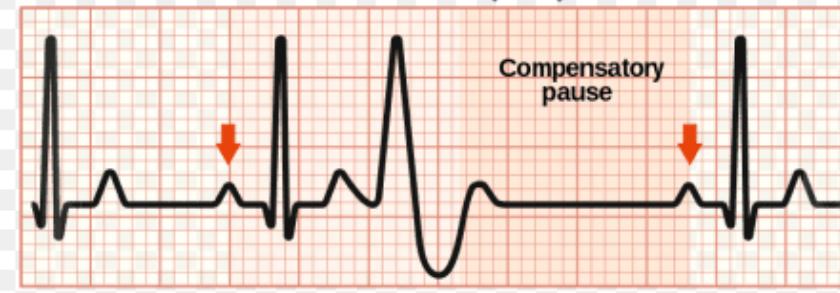
A : انقباض زودرس دهليزي
Premature Atrial Contraction (PAC)

Premature atrial contraction (PAC)



V : انقباض زودرس بطنی
Premature Ventricular Contraction (PVC)

Premature ventricular contraction (PVC)



Normal sinus rhythm



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



filenames

```
['100.csv',
'100annotations.txt',
'101.csv',
'101annotations.txt',
'102.csv',
'102annotations.txt',
'103.csv',
'103annotations.txt',
'104.csv',
'104annotations.txt',
'105.csv',
'105annotations.txt',
'106.csv',
'106annotations.txt',
'107.csv',
'107annotations.txt',
'108.csv',
'108annotations.txt',
'109.csv',
'109annotations.txt',
'110.csv',
'110annotations.txt',
'111.csv',
'111annotations.txt',
'112.csv',
'112annotations.txt',
'113.csv',
'113annotations.txt',
'114.csv',
'114annotations.txt',
'115.csv',
'115annotations.txt',
'116.csv',
'116annotations.txt',
'117.csv',
'117annotations.txt',
'118.csv',
```

سیگنال خام در قالب فایل های CSV و عنوان کلاس آریتمی برای هر سیگنال در قالب فایل های txt را در لیستی به نام Annotations ورودی های ما از پایگاه داده هستند



مقدمه پژوهش

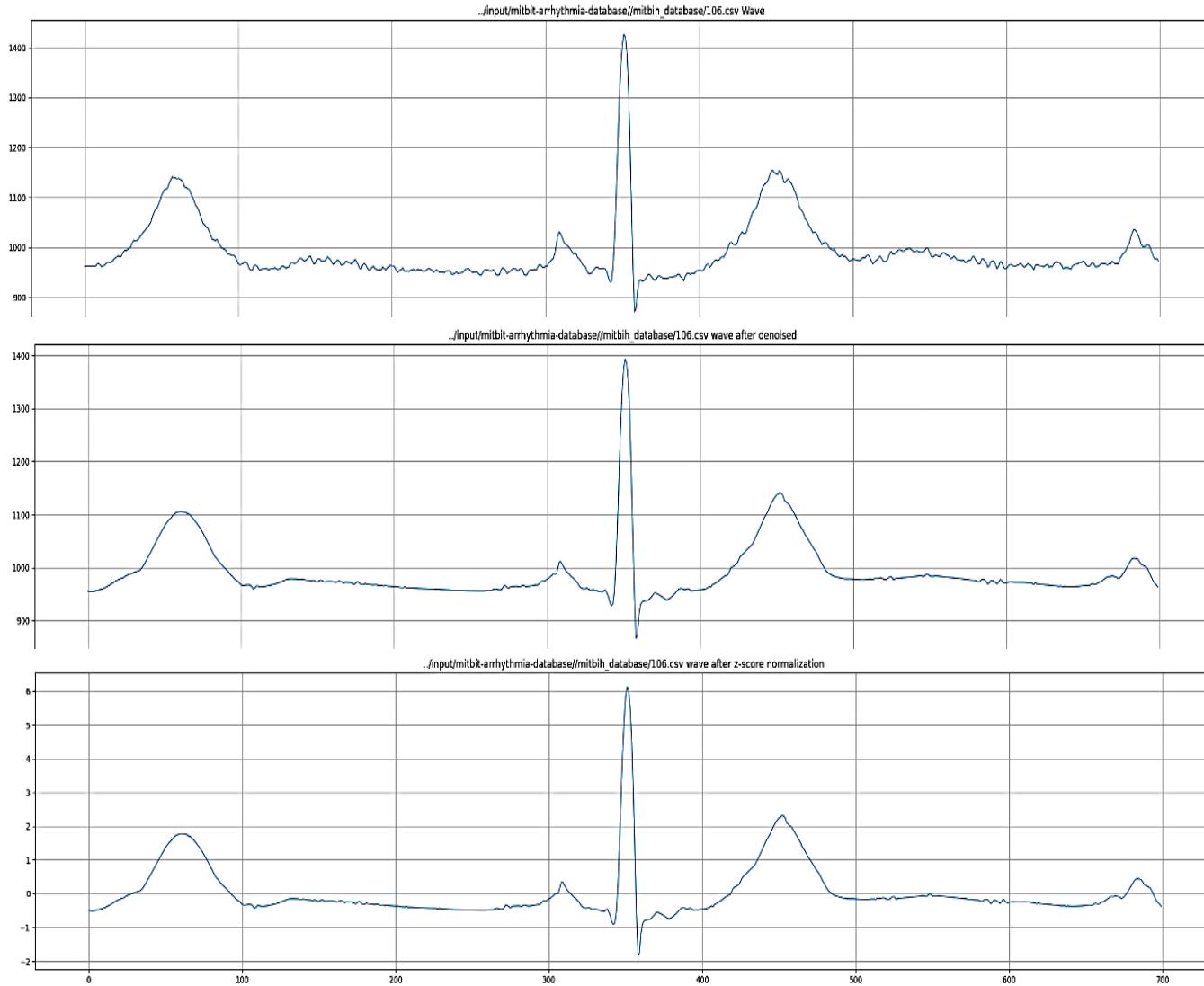
بررسی و پردازش
مجموعه داده

روش شناسی



نتیجه گیری





۱

پیش از حذف نویز

۲

پس از حذف نویز

۳

پس از حذف نویز و نرمال سازی

مقدمه پژوهش



بررسی و پردازش مجموعه داده



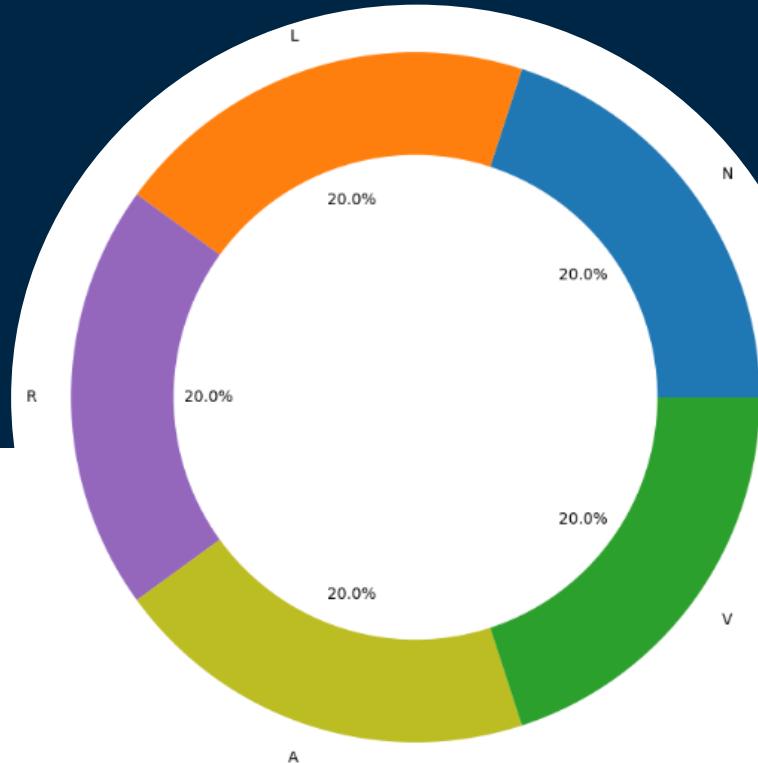
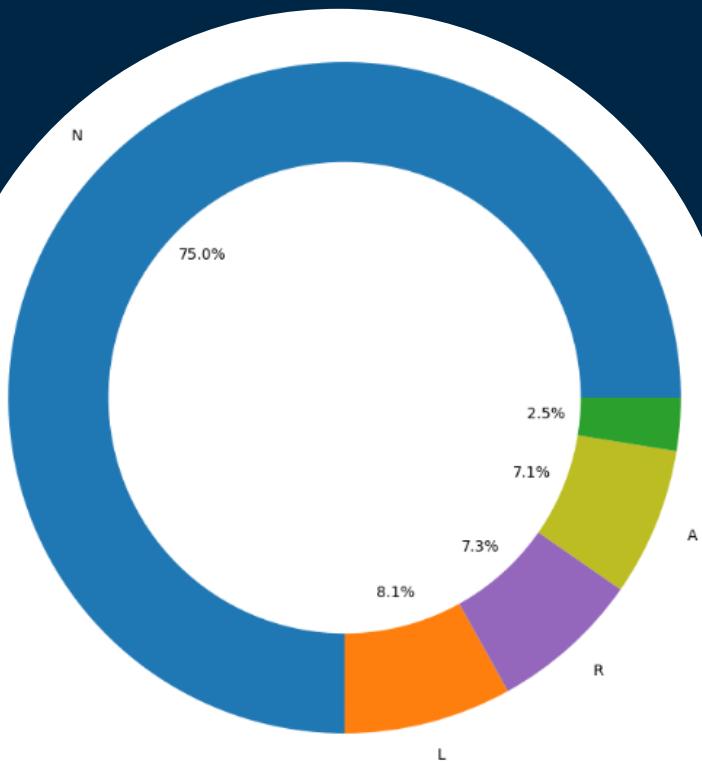
روش شناسی



نتیجه گیری



چگونگی توزیع داده ها در کلاس های مختلف آریتمی قبل و بعد از متعادل سازی



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



تشخیص آریتمی های قلبی به کمک الگوریتم شبکه عصبی مصنوعی



شبکه عصبی مصنوعی از ساختار مغز و شبکه‌های عصبی طبیعی الهام گرفته شده است. در سلول عصبی مصنوعی، داده‌های ورودی‌ها با ضرب در وزن آن‌ها و اعمال توابع محاسباتی (انواع تابع هزینه و فعال‌سازی) مقدار خروجی را تعیین می‌کند.

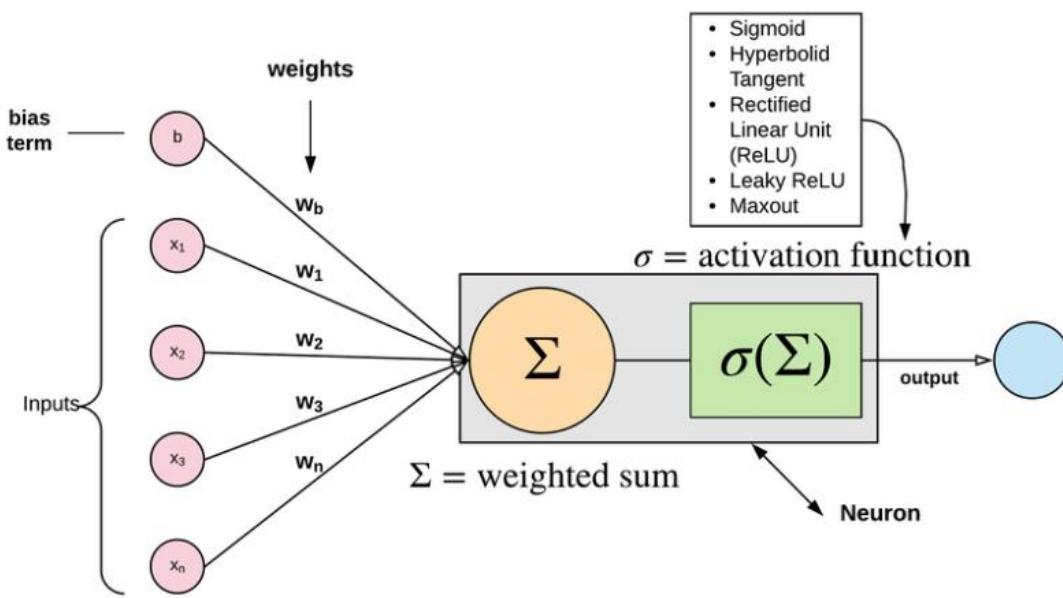
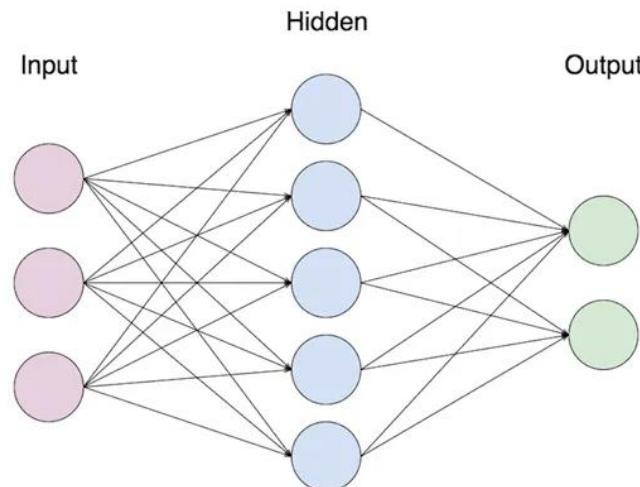
مقدمه پژوهش

بررسی و پردازش
مجموعه داده

تشخیص با ANN



نتیجه گیری



محاسبات بهینه سازی می تواند فرآیندی باشد به ما کمک کند با تغییر وزن‌های شبکه تابع اتلاف صفر شود. برای رسیدن به راه حل بهینه فرایند انجام آزمایش به صورت تکراری انجام می‌شود تا با هر بار تکرار اتلاف کمتر شده و به صفر نزدیک شود.

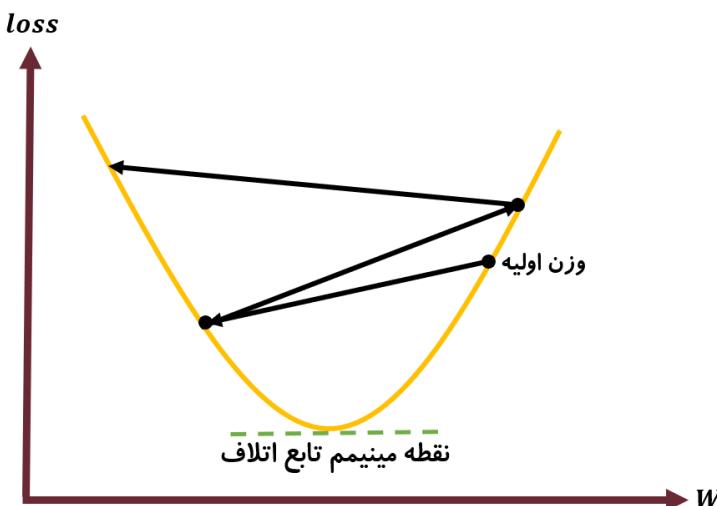
مقدمه پژوهش

بررسی و پردازش
مجموعه داده

تشخیص با ANN



نتیجه گیری



مدل سازی و آموزش شبکه عصبی مصنوعی

```
# Instantiate an empty model creates an empty linear stack of layers for building the model
ANNmodel = Sequential() # Dombale Tavali

# input layer of the model have an input shape of 360
# Input shape = N = 360

# regularization is a technique used to prevent overfitting (don't make your weights too big!)
# additional term added to the loss function during the training of a machine learning model

# Hidden Layer (Dense Layer with ReLU Activation) adds a dense (fully connected) layer to the model with 128 units
ANNmodel.add(Dense(128, bias_regularizer=regularizers.l2(0.0001), input_shape=(360,), activation='relu'))

# Output Layer (Dense Layer with Softmax Activation) adds the output layer to the model with 5 units
ANNmodel.add(Dense(5, bias_regularizer=regularizers.l2(0.0001)))

#used for multi-class classification
ANNmodel.add(Softmax())

history = ANNmodel.fit(train_x, train_y, batch_size=1000, epochs=20, verbose=1, validation_data=(test_x, test_y))

ANNmodel.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
```

Indicator variable

$$-\sum_{j=1}^M y_j \log(p(y_j))$$

Prob of class j

Sum over trials

$$-\sum_{i=1}^N y_i \log(p(y_i)) + (1 - y_i) \log(1 - p(y_i))$$

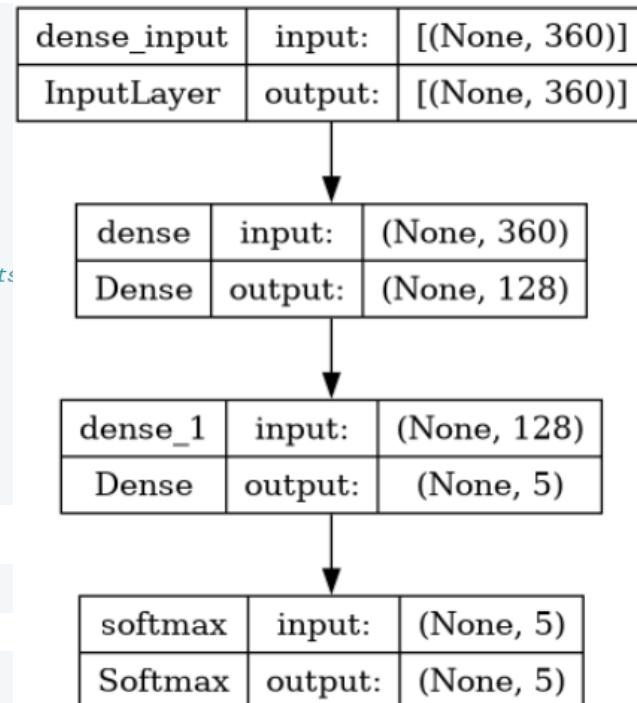
Label

Prob of positive class

Sum over classes

Label

Prob of positive class



مقدمه پژوهش



بررسی و پردازش
مجموعه داده

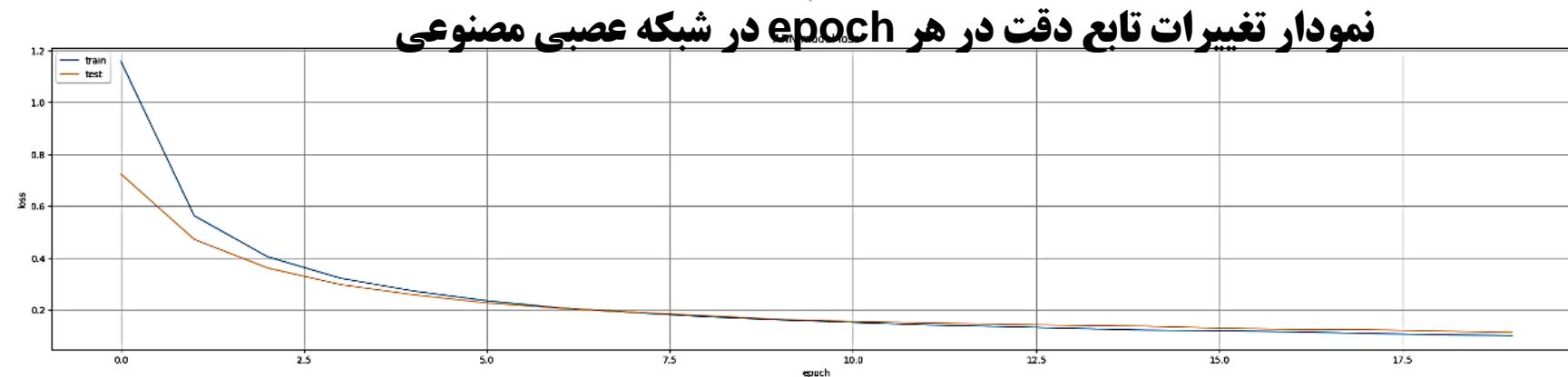
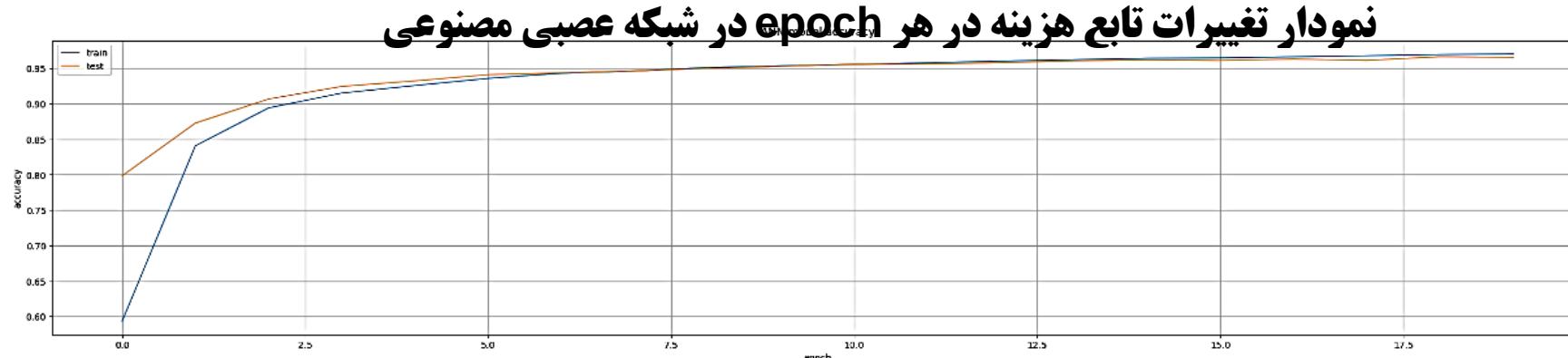


تشخیص با



نتیجه گیری





خلاصه ای از نتایج شبکه عصبی مصنوعی

دقت مدل	تعداد کلاس آریتمی	صحت	F1 معیار	بازیابی	مدت زمان آموزش الگوریتم	الگوریتم یادگیری ماشین
٪۹۶,۵۶	۵ کلاس	۰,۹۶۵۵۰	۰,۹۶۵۵۳	۰,۹۶۵۹	۴,۱۱ ثانیه	شبکه عصبی مصنوعی

مقدمه پژوهش

بررسی و پردازش
مجموعه داده

تشخیص با ANN



نتیجه گیری

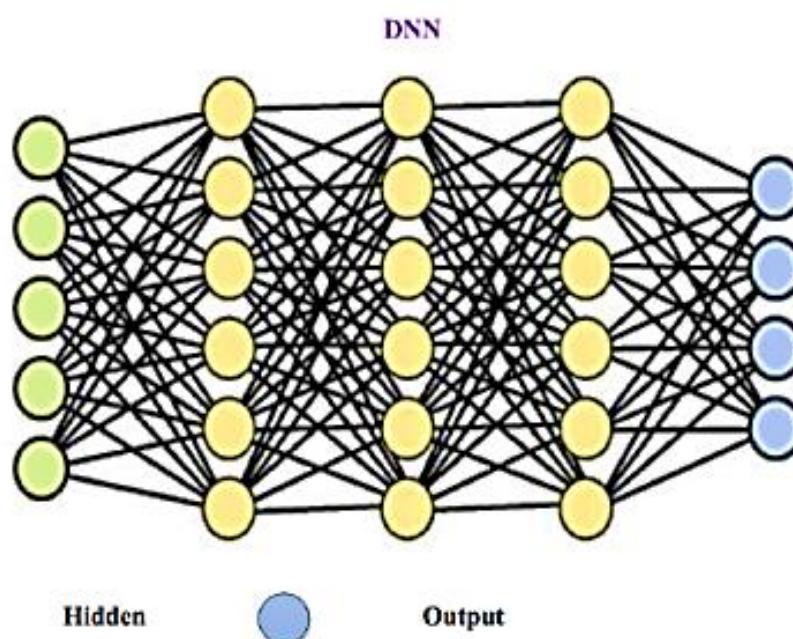
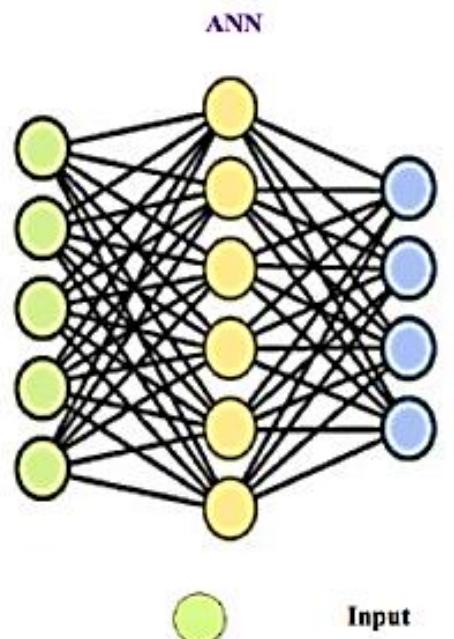


تشخیص آریتمی های قلبی به کمک آلگوریتم شبکه عصبی عمیق



معماری‌های شبکه عصبی عمیق به عنوان مجموعه‌ای از الگوریتم‌های یادگیری ماشین تعریف می‌شوند که به صورت سلسله مراتبی یاد می‌گیرند و شامل چندین لایه هستند به گونه‌ای که یادگیری در لایه‌های بالاتر توسط و بر پایه یادگیری آماری که در لایه‌های سطح پایین‌تر اتفاق می‌افتد، تعریف می‌شود.

شبکه عصبی مصنوعی از یک یا دو لایه پنهان برای پردازش داده استفاده می‌کند، در حالی که شبکه عصبی عمیق عمدتاً از چندین لایه بین لایه‌های ورودی و خروجی استفاده می‌کند و پیچیده‌تر از سایر شبکه‌های عصبی است زیرا دارای لایه‌های بیشتری نسبت به آنها است



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



تشخیص با DNN



نتیجه گیری





مدل سازی و آموزش شبکه عصبی عمیق

```
# Instantiate an empty model
DNNmodel = Sequential()

# Input shape = N = 360
# regularization is a technique used to prevent overfitting (don't make your weights too big!)
# additional term added to the loss function during the training of a machine learning model

# Hidden Layers (Dense Layer with ReLU Activation) adds a dense (fully connected) layer to the model
DNNmodel.add(Dense(512, bias_regularizer=regularizers.l2(0.0001), input_shape=(360,), activation='relu'))
DNNmodel.add(Dense(1024, bias_regularizer=regularizers.l2(0.0001), activation='relu'))
DNNmodel.add(Dense(512, bias_regularizer=regularizers.l2(0.0001), activation='relu'))
DNNmodel.add(Dense(128, bias_regularizer=regularizers.l2(0.0001), activation='relu'))

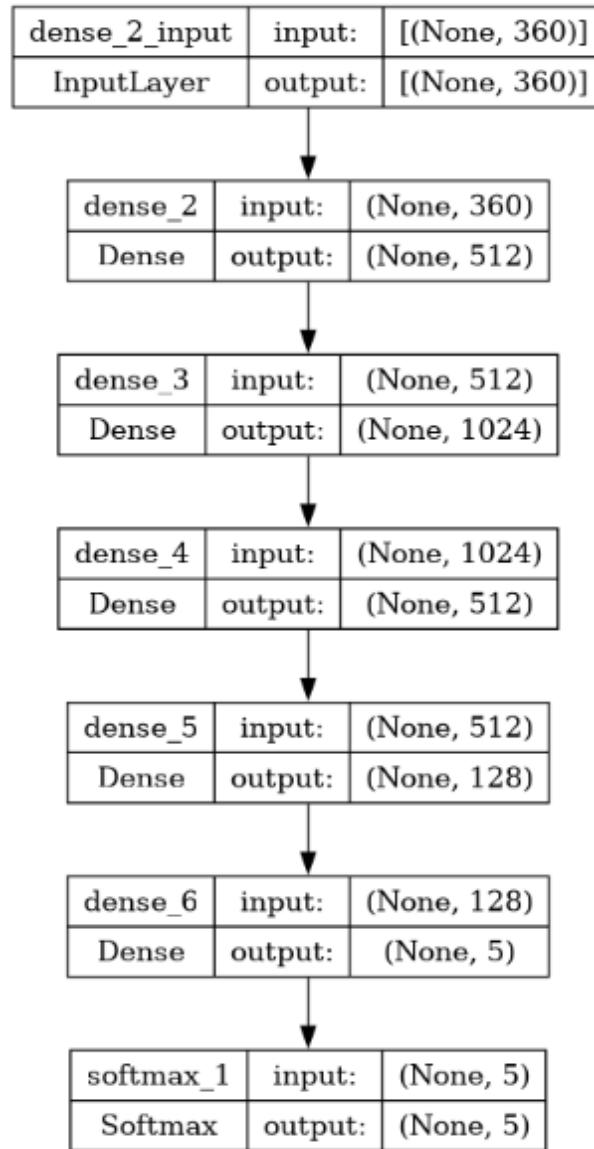
# Output Layer (Dense Layer with Softmax Activation) adds the output layer to the model with 5 units
DNNmodel.add(Dense(5, bias_regularizer=regularizers.l2(0.0001)))

#used for multi-class classification
DNNmodel.add(Softmax())

# number of trainable parameters, the input and output shapes of each layer, and the total number of parameters
DNNmodel.summary()

DNNmodel.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])

history = DNNmodel.fit(train_x, train_y, batch_size=1000, epochs=20, verbose=1, validation_data=(test_x, t
```



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



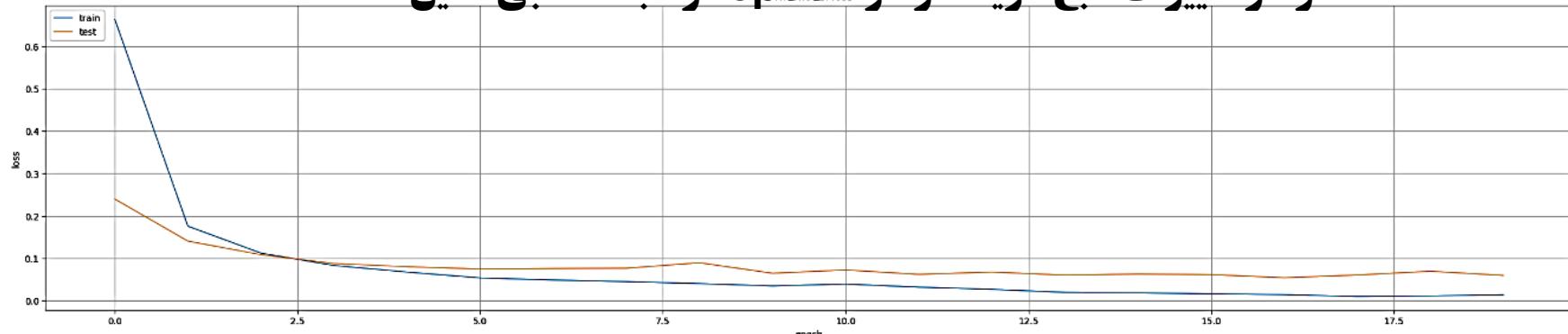
تشخیص با DNN



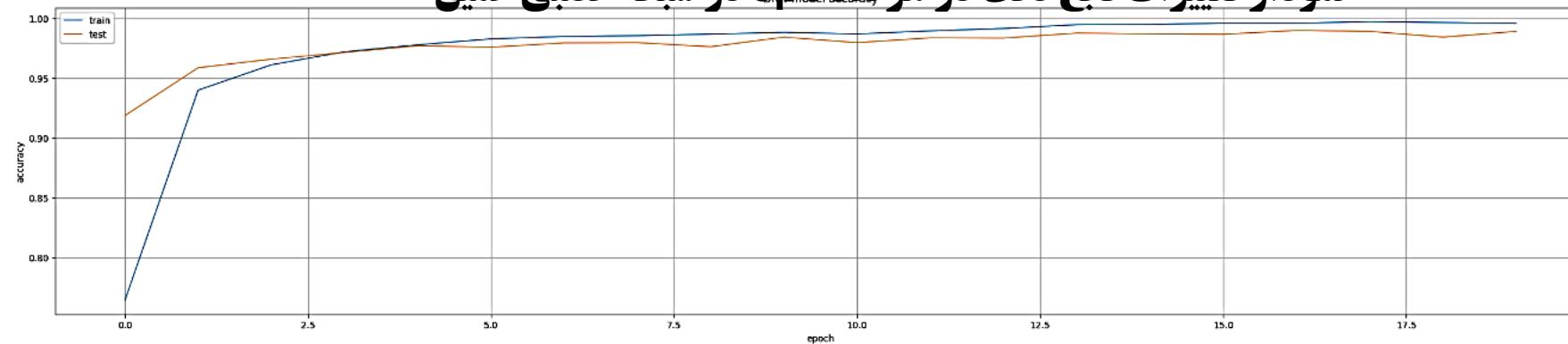
نتیجه گیری



نمودار تغییرات تابع هزینه در هر epoch در شبکه عصبی عمیق



نمودار تغییرات تابع دقت در هر epoch در شبکه عصبی عمیق



خلاصه ای از نتایج شبکه عصبی عمیق

دقت مدل	تعداد کلاس آریتمی	صحت	F1 معیار	بازیابی	مدت زمان آموزش الگوریتم	الگوریتم یادگیری ماشین
%۹۸,۹۰	۵ کلاس	۰,۹۸۸۹۵	۰,۹۸۸۹۵	۰,۹۸۸۹۶	۲۷,۷ ثانیه	شبکه عصبی عمیق

مقدمه پژوهش

بررسی و پردازش
مجموعه داده

تشخیص با DNN



نتیجه گیری

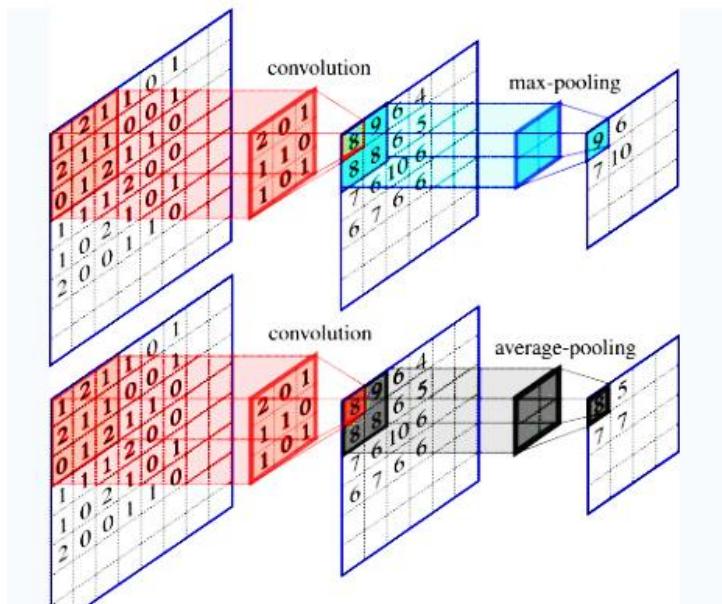
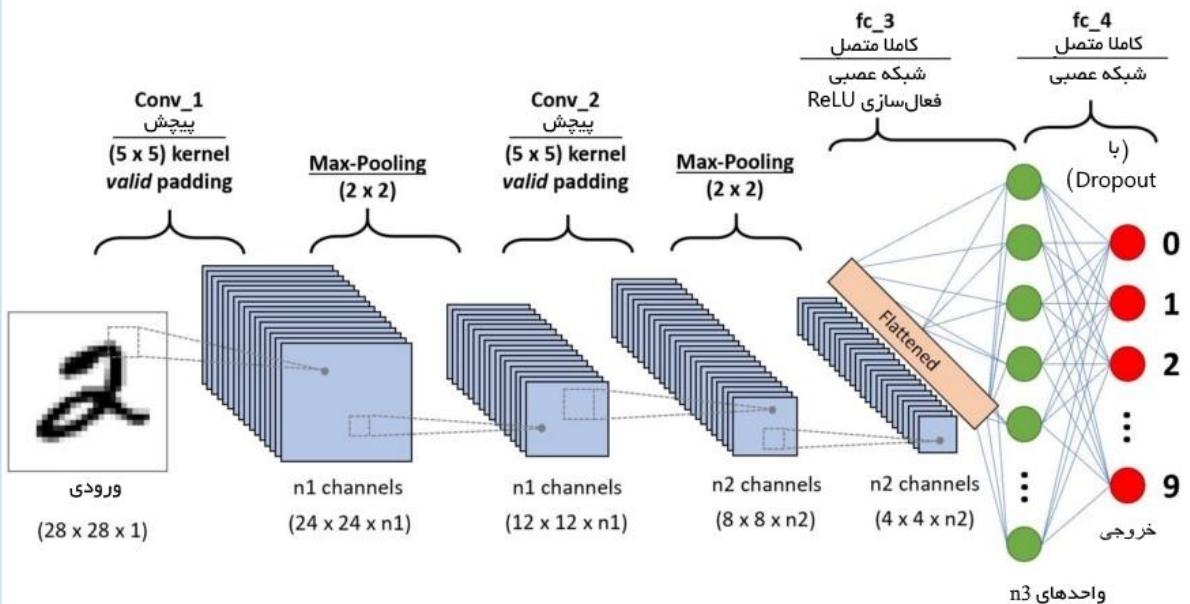


تشخیص آریتمی های قلبی به کمک الگوریتم شبکه عصبی کانولوشن



شبکه عصبی کانولوشن یکی از مهم ترین شبکه های عصبی عمیق است که از بخش بینایی مغز الگو برداری شده است. دسته های کوچکی از نورون های بینایی که در هر قسمت به تنها یی مشغول هستند باعث ایجاد شبکه های به هم پیوسته و دیدن یک منطقه میشوند. شبکه عصبی کانولوشن از لایه هی زیر تشکیل شده است:

// fully connected // input fully connected // pooling // لایه
لایه output fully connected



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



تشخیص با



نتیجه گیری



```
# Instantiate an empty model
CNNmodel = Sequential()

# Adding a Convolution Layer C1
# Input shape = N = (360 x 1)
# No. of filters = 16 : Each filter is responsible for learning different features from the input
# Filter size = f = (13 x 1) : the region of the input data that a single filter "sees"
# Padding = true : padding means the output size will be the same as the input size (add with zeros if necessary)
# Strides = S = 1

# N=length of the input sequence // f=filter size (kernel size) // P=the padding size // S=stride
# Size of each feature map in C1 is (N-f+2P)/S + 1 = 360-13+2*6 +1 = 360
# No. of parameters between input layer and C1 = [kernal size(weight) * input channel + 1(bias)] * filters = (13*1 + 1)*16 = 224
CNNmodel.add(Conv1D(filters=16, kernel_size=13, padding='same', activation='relu', input_shape=(360, 1)))

# Adding an Average Pooling Layer S2
# Input shape = N = (360 x 16)
# No. of filters = 16
# Filter size = f = (3 x 1) : computes the average value within windows of size 3
# Padding = P = 0
# Strides = S = 2
# Size of each feature map in S2 is (N-f+2P+1)/S = (360-3+0+1)/2 = 179
# No. of parameters between C1 and S2 = (1+1)*16 = 32
CNNmodel.add(AvgPool1D(pool_size=3, strides=2))

CNNmodel.add(Conv1D(filters=32, kernel_size=15, padding='same', activation='relu'))
CNNmodel.add(AvgPool1D(pool_size=3, strides=2))

CNNmodel.add(Conv1D(filters=64, kernel_size=17, padding='same', activation='relu'))
CNNmodel.add(AvgPool1D(pool_size=3, strides=2))

CNNmodel.add(Conv1D(filters=128, kernel_size=19, padding='same', activation='relu'))
CNNmodel.add(AvgPool1D(pool_size=3, strides=2))

CNNmodel.add(Flatten())
CNNmodel.add(Dropout(0.5))

CNNmodel.add(Dense(35, kernel_regularizer=regularizers.l2(0.0001), bias_regularizer=regularizers.l2(0.0001)))
CNNmodel.add(Dense(5, kernel_regularizer=regularizers.l2(0.0001), bias_regularizer=regularizers.l2(0.0001)))

CNNmodel.add(Softmax())
| CNNmodel.summary()

CNNmodel.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
history = CNNmodel.fit(train_x, train_y, batch_size=1000, epochs=20, verbose=1, validation_data=(test_x, test_y))
```

مدل سازی و آموزش شبکه عصبی کانولوشن

Model: "sequential_12"

Layer (type)	Output Shape	Param #
conv1d_40 (Conv1D)	(None, 360, 16)	224
average_pooling1d_40 (AveragePooling1D)	(None, 179, 16)	0
conv1d_41 (Conv1D)	(None, 179, 32)	7712
average_pooling1d_41 (AveragePooling1D)	(None, 89, 32)	0
conv1d_42 (Conv1D)	(None, 89, 64)	34880
average_pooling1d_42 (AveragePooling1D)	(None, 44, 64)	0
conv1d_43 (Conv1D)	(None, 44, 128)	155776
average_pooling1d_43 (AveragePooling1D)	(None, 21, 128)	0
flatten_10 (Flatten)	(None, 2688)	0
dropout_10 (Dropout)	(None, 2688)	0
dense_27 (Dense)	(None, 35)	94115
dense_28 (Dense)	(None, 5)	180
softmax_12 (Softmax)	(None, 5)	0
<hr/>		
Total params: 292887 (1.12 MB)		
Trainable params: 292887 (1.12 MB)		
Non-trainable params: 0 (0.00 Byte)		

مقدمه پژوهش



بررسی و پردازش
مجموعه داده

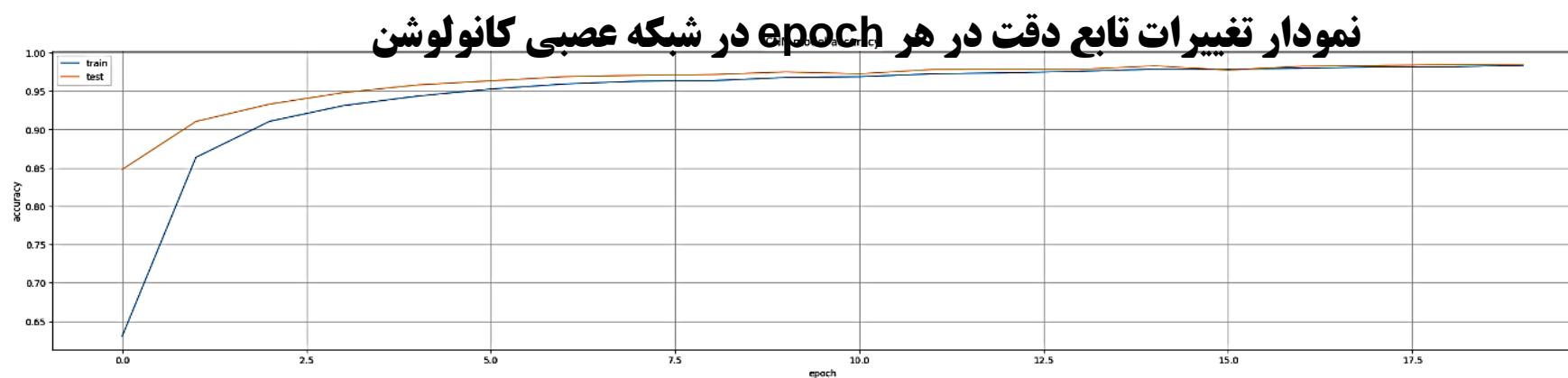
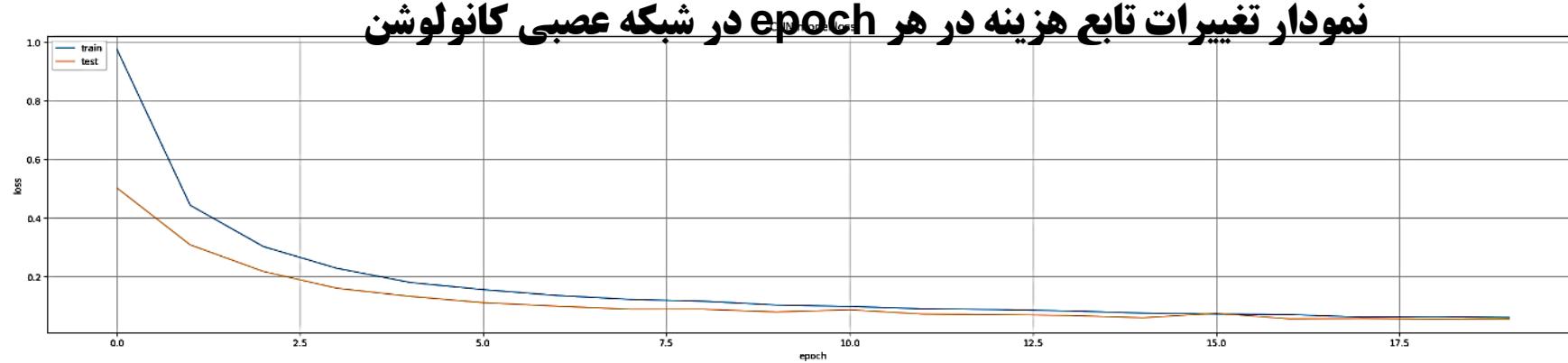


تشخیص با CNN



نتیجه گیری





خلاصه ای از نتایج شبکه عصبی کانولوشن

دقت مدل	تعداد کلاس آریتمی	صحت	F1 معیار	بازیابی	مدت زمان آموزش الگوریتم	الگوریتم یادگیری ماشین
٪۹۸.۴۴	۵ کلاس	۰.۹۸۴۱	۰.۹۸۴۱	۰.۹۸۴۲	۳۵۹.۴ ثانیه	شبکه عصبی کانولوشن

مقدمه پژوهش

بررسی و پردازش
مجموعه داده

تشخیص با CNN



نتیجه گیری



تشخیص آریتمی های قلبی به کمک الگوریتم K نزدیک ترین همسایه



این الگوریتم برای مسائل طبقه‌بندی k نزدیک ترین همسایه را پیدا و با اکثریت آرا در نزدیک‌ترین همسایگان کلاس را پیش‌بینی می‌کند.

در ابتدا مقدار K را تعیین می‌کنیم که همان تعداد نزدیک‌ترین همسایه‌ها هستند. فاصله‌ی میان نمونه مورد آزمایش را با نمونه داده‌های آموزشی موجود محاسبه می‌کنیم.

K تا از نزدیک‌ترین همسایگان را انتخاب می‌کنیم و بر اساس برچسب این K نمونه نمونه‌ی جدید را برچسب گذاری می‌کنیم.

مقدمه پژوهش



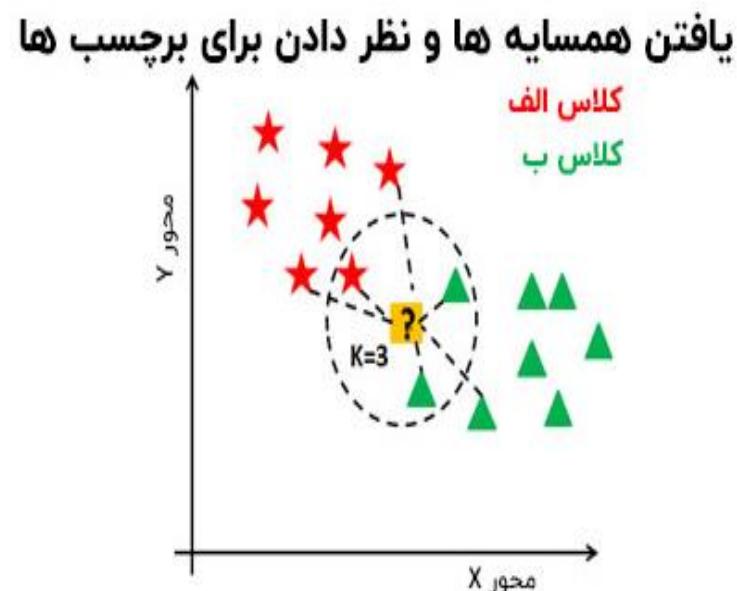
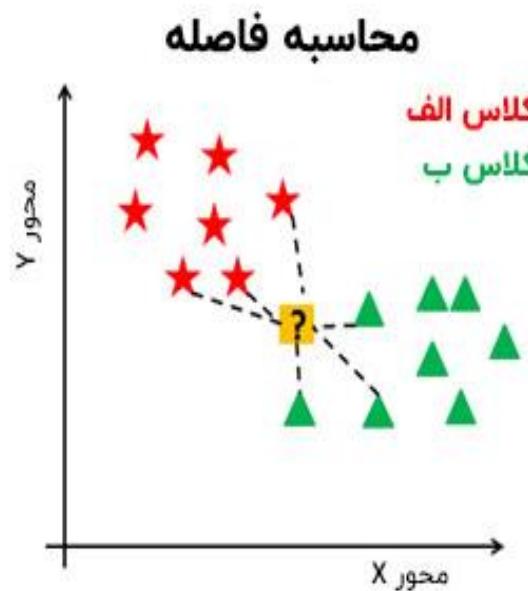
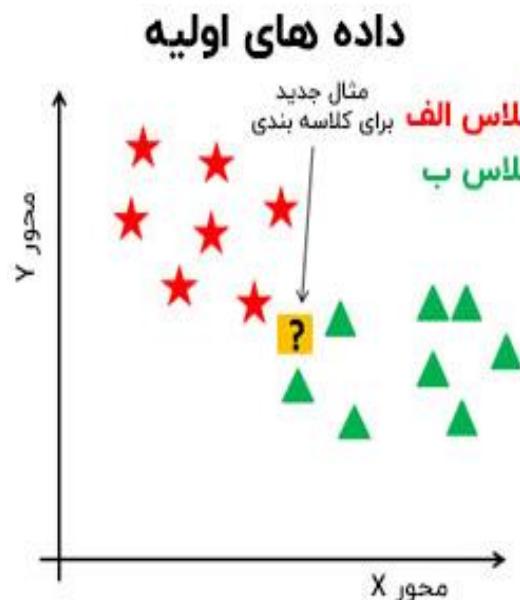
بررسی و پردازش
مجموعه داده



تشخیص با KNN



نتیجه گیری



مدل سازی و آموزش الگوریتم K نزدیک ترین همسایه

```
from sklearn.neighbors import KNeighborsClassifier

# Creating the KNN Model
# when the model is asked to classify a new data point, it looks at the labels of the three training instances that are closest
# to the given data point and assigns the class label that is most common among these three neighbors
KNNmodel = KNeighborsClassifier(n_neighbors=3)

# Training the Model
# d2_train_x represents the input features, and d1_train_y represents the corresponding class labels
KNNmodel.fit(d2_train_x,d1_train_y)
```

KNeighborsClassifier

KNeighborsClassifier(n_neighbors=3)

```
# predicts the class labels for the test set (d2_test_x) using the k-Nearest Neighbors (KNN)
KNNpred = KNNmodel.predict(d2_test_x)
```

```
print(metrics.accuracy_score(d1_test_y, KNNpred))
```

0.972

خلاصه ای از نتایج الگوریتم K نزدیک ترین همسایه

دقت مدل	تعداد کلاس آریتمی	F1	متغیر همسایه	مدت زمان آموزش مربوط به قبایل مراحتی و آمور معیار الگوریتم	الگوریتم	الگوریتم یادگیری ماشین
۰.۹۷۲	۵ کلاس	۰.۹۷	۰.۹۷	۰.۹۷	۱۱ ثانیه	الگوریتم K نزدیک ترین همسایه

مقدمه پژوهش



بررسی و پردازش
مجموعه داده



تشخیص با KNN



نتیجه گیری



تشخیص آریتمی های قلبی به کمک الگوریتم درخت تصمیم

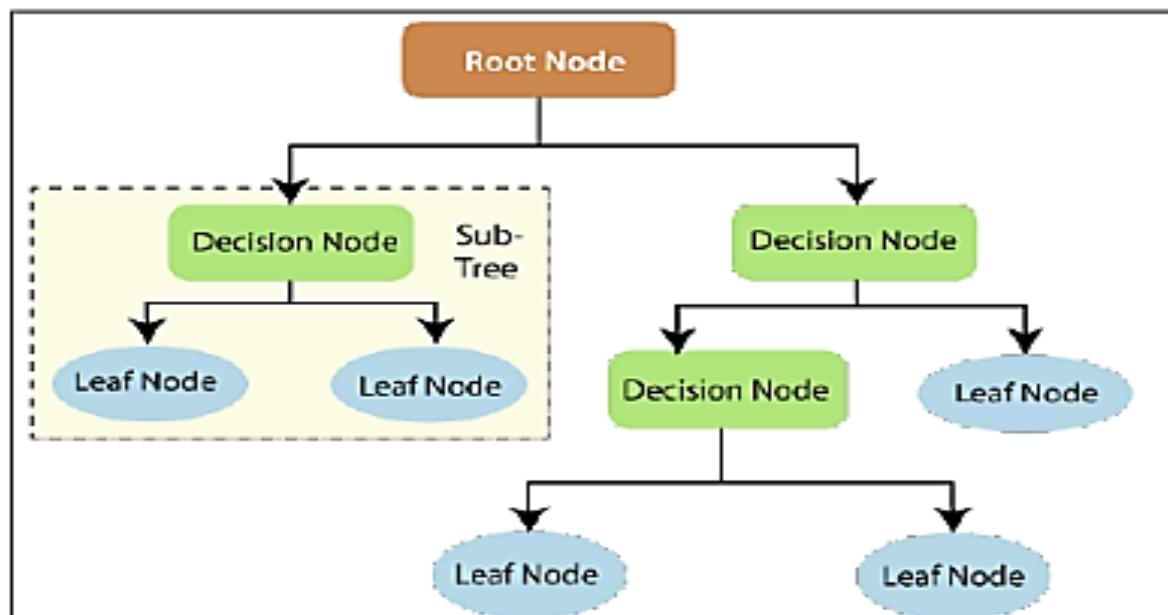


هر درخت از گره ریشه و گرههای داخلی و شاخهها و گرههای برگ تشکیل شده است،
گره ریشه، نقطه شروع فرآیند تصمیم‌گیری را نشان می‌دهد.
گرههای داخلی مراحل میانی در فرآیند تصمیم‌گیری را نشان می‌دهند.
شاخهها خطوطی هستند که گرهها را به یکدیگر متصل می‌کنند.
گرههای برگ، نشان‌دهنده نتیجه نهایی فرآیند تصمیم‌گیری هستند.

مقدمه پژوهش

بررسی و پردازش
مجموعه دادهتشخیص با
درخت تصمیم

نتیجه گیری



مدل سازی و آموزش الگوریتم درخت تصمیم

```
from sklearn import tree

# the randomness is reproducible. That is, if you run the same code with the same random_state, you should get the same results
# make the training process reproducible
dt = tree.DecisionTreeClassifier(random_state=0)

# Training the Decision Tree : d2_train_x as features and d1_train_y as labels
dt.fit(d2_train_x, d1_train_y)

# Making Predictions : test data (d2_test_x)
DTpred = dt.predict(d2_test_x)
```

```
# compares true labels with predicted labels
print(metrics.accuracy_score(d1_test_y, DTpred))
```

0.9452

خلاصه ای از نتایج الگوریتم درخت تصمیم

دقت مدل	تعداد کلاس آریتمی	صحت	F1 معیار	بازیابی	مدت زمان آموزش الگوریتم	الگوریتم یادگیری ماشین
٪۹۴,۵	۵ کلاس	۰,۹۵	۰,۹۵	۰,۹۵	۱۵ ثانیه	الگوریتم درخت تصمیم

مقدمه پژوهش

بررسی و پردازش
مجموعه دادهتشخیص با
درخت تصمیم

نتیجه گیری



تشخیص آریتمی های قلبی به کمک الگوریتم جنگل های تصادفی

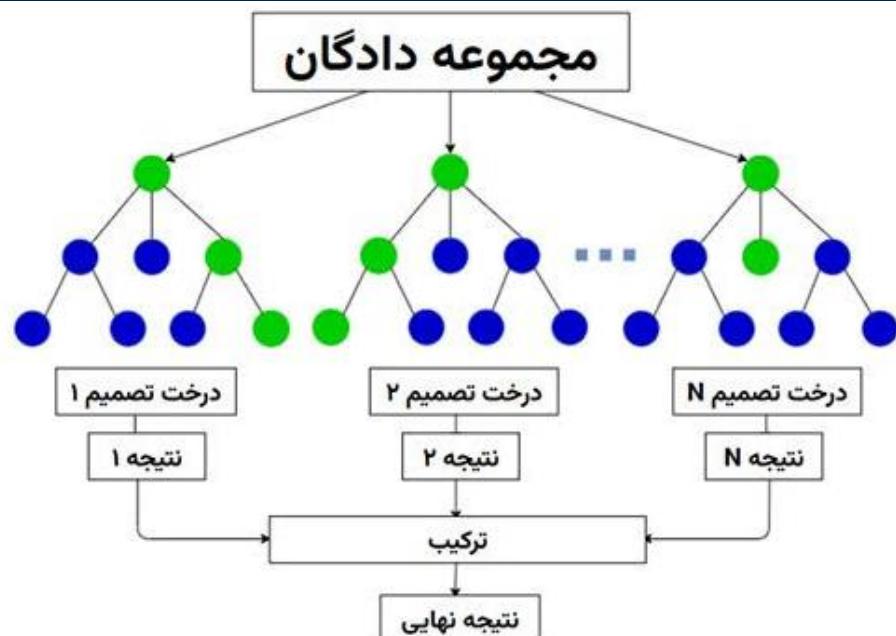


جنگل‌های تصادفی، یک مجموعه از درخت‌های تصمیم هستند که به طور معمول عملکرد بهتری نسبت به یک درخت تصمیم مستقل دارند. در این روش داده‌ها به‌طور موازی و مستقل از یکدیگر آموزش داده می‌شوند و در نهایت با استفاده از یک فرآیند میانگین‌گیری، نمونه‌ها با هم ترکیب می‌شوند. به صورتی که، جنگل تصادفی از چندین درخت تصمیم ساخته شده است

مقدمه پژوهش

بررسی و پردازش
مجموعه دادهتشخیص با
جنگل‌های تصادفی

نتیجه گیری



مدل سازی و آموزش الگوریتم جنگل تصادفی

```
from sklearn.ensemble import RandomForestClassifier

# number of trees in the forest : Random forest will consist of 100 decision trees
rf = RandomForestClassifier(n_estimators = 100)

# Training the RandomForest : d2_train_x as features and d1_train_y as labels
rf.fit(d2_train_x, d1_train_y)
```

RandomForestClassifier
RandomForestClassifier()

```
# Making Predictions : test data (d2_test_x)
RFpred = rf.predict(d2_test_x)
```

```
# compares true labels with predicted labels
print(metrics.accuracy_score(d1_test_y, RFpred))
```

0.9818

خلاصه ای از نتایج الگوریتم جنگل تصادفی

دقت مدل	تعداد کلاس آرینتی	صحت	F1 معیار	بازیابی	مدت زمان آموزش الگوریتم	الگوریتم یادگیری ماشین
%۹۸,۱۸	۵ کلاس	۰,۹۸	۰,۹۸	۰,۹۸	۴۱ ثانیه	الگوریتم جنگل تصادفی

مقدمه پژوهش

بررسی و پردازش
مجموعه دادهتشخیص با
جنگل‌های تصادفی

نتیجه گیری



مقایسه و نتیجه گیری



نتیجه گیری



دقت مدل	تعداد کلاس آریتمی	صحت	F1 معیار	بازیابی	مدت زمان آموزش الگوریتم	الگوریتم یادگیری ماشین
٪۹۶,۵۶	۵ کلاس	۰,۹۶۵۵۰	۰,۹۶۵۵۳	۰,۹۶۵۹	۴,۱۱ ثانیه	شبکه عصبی مصنوعی
٪۹۸,۹۰	۵ کلاس	۰,۹۸۸۹۵	۰,۹۸۸۹۵	۰,۹۸۸۹۶	۲۷,۷ ثانیه	شبکه عصبی عمیق
٪۹۸,۴۴	۵ کلاس	۰,۹۸۴۱	۰,۹۸۴۱	۰,۹۸۴۲	۳۵۹,۴ ثانیه	شبکه عصبی کانولوشن
٪۹۷,۲	۵ کلاس	۰,۹۷	۰,۹۷	۰,۹۷	۱۱ ثانیه	الگوریتم K نزدیک ترین همسایه
٪۹۴,۵	۵ کلاس	۰,۹۵	۰,۹۵	۰,۹۵	۱۵ ثانیه	الگوریتم درخت تصمیم
٪۹۸,۱۸	۵ کلاس	۰,۹۸	۰,۹۸	۰,۹۸	۴۱ ثانیه	الگوریتم جنگل تصادفی

مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



چند نمونه از بررسی دقیقیت الگوریتم بر روی پایگاه داده

```

import random
i = random.randint(0, len(test_x)-1)

output = CNNmodel(np.expand_dims(test_x[i], 0))

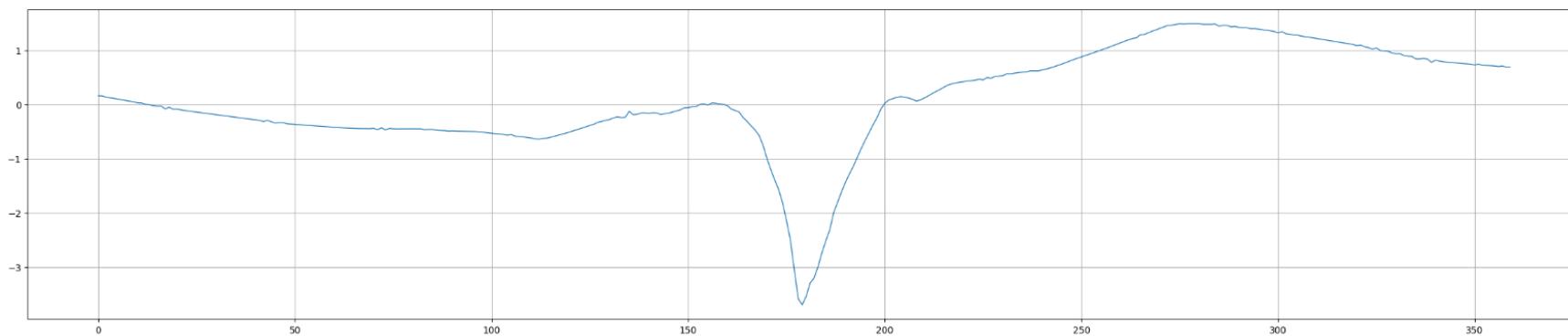
pred = output.numpy()[0]

plt.plot(test_x[i])

print("Actual label: ", classes[np.argmax(test_y[i])])
print("Model prediction : ", classes[np.argmax(pred)], " with probability ", pred[np.argmax(pred)])

```

Actual label: L
 Model prediction : L with probability 0.9999974



مقدمه پژوهش

بررسی و پردازش
مجموعه داده

روش شناسی



نتیجه گیری



چند نمونه از بررسی دقیقیت الگوریتم بر روی پایگاه داده

```

import random
i = random.randint(0, len(test_x)-1)

output = ANNmodel(np.expand_dims(test_x[i], 0))

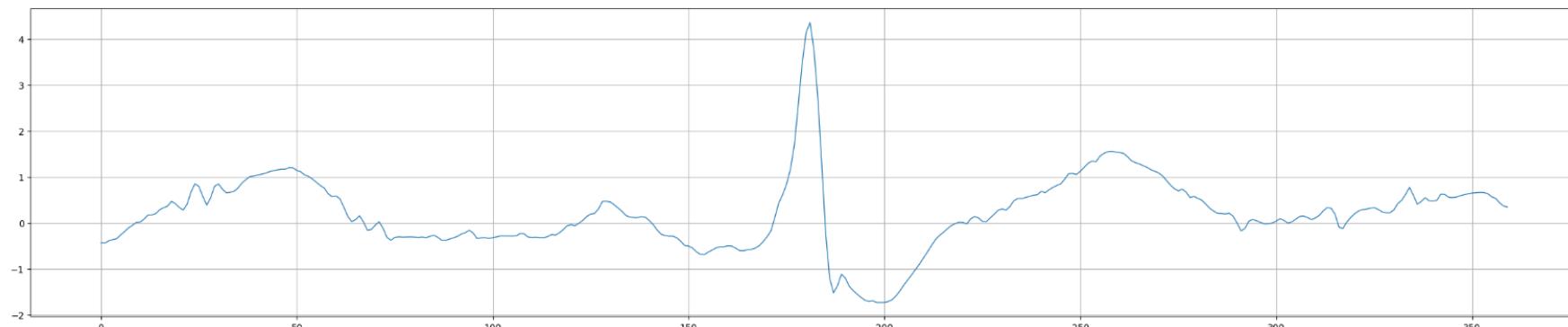
pred = output.numpy()[0]

plt.plot(test_x[i])

print("Actual label: ", classes[np.argmax(test_y[i])])
print("Model prediction : ", classes[np.argmax(pred)], " with probability ", pred[np.argmax(pred)])

```

Actual label: R
 Model prediction : R with probability 0.99941635



مقدمه پژوهش

بررسی و پردازش
مجموعه داده

روش شناسی



نتیجه گیری



چند نمونه از بررسی دقیقیت الگوریتم بر روی پایگاه داده

```

import random
i = random.randint(0, len(test_x)-1)

output = DNNmodel(np.expand_dims(test_x[i], 0))

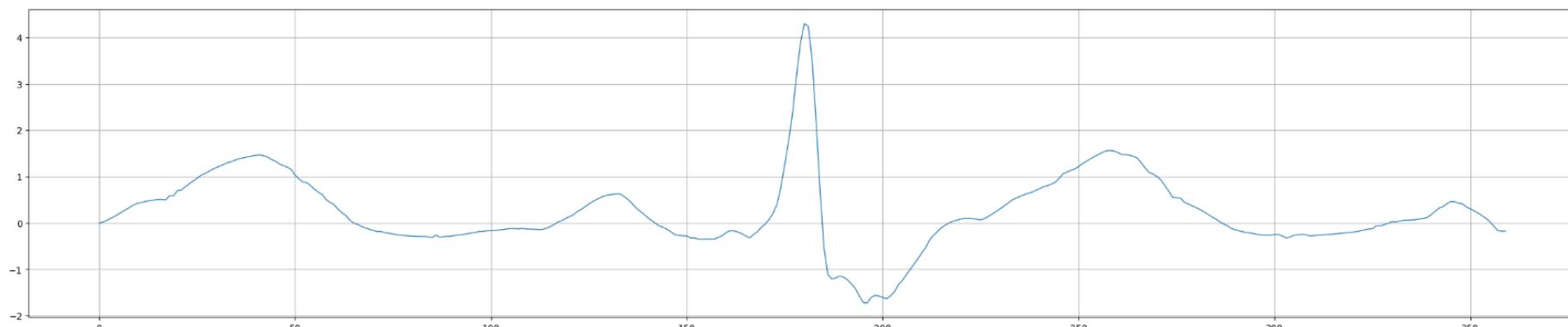
pred = output.numpy()[0]

plt.plot(test_x[i])

print("Actual label: ", classes[np.argmax(test_y[i])])
print("Model prediction : ", classes[np.argmax(pred)], " with probability ", pred[np.argmax(pred)])

```

Actual label: R
 Model prediction : R with probability 1.0



مقدمه پژوهش



بررسی و پردازش
مجموعه داده



روش شناسی



نتیجه گیری



پسرانہ توجہ شما

