**HEART DISEASES PREDICTION**

**import** pandas **as** pd

heart **=** pd.read\_csv('heart.csv')

heart.head()

heart.columns

cols\_to\_norm **=** ['age','gender','chest pain','sugar','restecg','exang','slope','ca','thal','bp','cholestrol','thalak','old peak','contactno']

heart[cols\_to\_norm] **=** heart[cols\_to\_norm].apply(**lambda** x: (x **-** x.min()) **/** (x.max() **-** x.min()))

*# heart.head()*

**import** tensorflow **as** tf

​

*#dis = tf.feature\_column.string\_column('disease')*

age **=** tf.feature\_column.numeric\_column('age')

gen **=** tf.feature\_column.numeric\_column('gender')

chest **=** tf.feature\_column.numeric\_column('chest pain')

sugar **=** tf.feature\_column.numeric\_column('sugar')

ecg **=** tf.feature\_column.numeric\_column('restecg')

ex **=** tf.feature\_column.numeric\_column('exang')

slop **=** tf.feature\_column.numeric\_column('slope')

ca **=** tf.feature\_column.numeric\_column('ca')

th **=** tf.feature\_column.numeric\_column('thal')

bp **=** tf.feature\_column.numeric\_column('bp')

chol **=** tf.feature\_column.numeric\_column('cholestrol')

thal **=** tf.feature\_column.numeric\_column('thalak')

old **=** tf.feature\_column.numeric\_column('old peak')

no **=** tf.feature\_column.numeric\_column('contactno')

**import** matplotlib.pyplot **as** plt

**%**matplotlib inline

heart['age'].hist(bins**=**20)

age\_buckets **=** tf.feature\_column.bucketized\_column(age, boundaries**=**[0.0,0.2,0.4,0.6,0.8,1.0])

feat\_cols **=** [age,gen,chest,sugar,ecg,ex,slop,th,ca,bp,chol,thal,old,no,age\_buckets]

**from** sklearn.tree **import** DecisionTreeClassifier

x\_data **=** heart.drop('contactno',axis**=**1)

labels **=** heart['contactno']

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x\_data,labels,test\_size**=**0.33, random\_state**=**101)

input\_func **=** tf.estimator.inputs.pandas\_input\_fn(x**=**X\_train,y**=**y\_train,batch\_size**=**10,num\_epochs**=**1000,shuffle**=True**)

model **=** tf.estimator.LinearClassifier(feature\_columns**=**feat\_cols,n\_classes**=**2)

​

input\_func **=** tf.estimator.inputs.pandas\_input\_fn(x**=**X\_train,y**=**y\_train,batch\_size**=**10,num\_epochs**=**1000)

model **=** tf.estimator.LinearClassifier(feature\_columns**=**feat\_cols,n\_classes**=**2)

model.train(input\_fn**=**input\_func,steps**=**1000)

​

pred\_input\_func **=** tf.estimator.inputs.pandas\_input\_fn(

x**=**X\_test,

batch\_size**=**10,

num\_epochs**=**1,

shuffle**=False**)

predictions **=** model.predict(pred\_input\_func)

list(predictions)

​

​