Heart Disease Prediction By Sabyasachi Bandyopadhyay

I. Importing essential libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')
```

```
['.ipynb_checkpoints', 'heart.csv', 'Heart_disease_prediction by sab
yasachiBandyopadhyay.ipynb', 'README.md', 'Untitled.ipynb']
```

II. Importing and understanding our dataset

In [3]:
```
dataset = pd.read_csv("heart.csv")
```

Verifying it as a 'dataframe' object in pandas

In [4]:
```
type(dataset)
```

Out[4]:
```
pandas.core.frame.DataFrame
```

Shape of dataset

In [5]:
```
dataset.shape
```

Out[5]:
```
(303, 14)
```

Printing out a few columns

In [6]:
```
dataset.head(5)
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [7]:
```
dataset.sample(5)
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
142	42	0	2	120	209	0	1	173	0	0.0	1	0	2	1
65	35	0	0	138	183	0	1	182	0	1.4	2	0	2	1
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3	0
21	44	1	2	130	233	0	1	179	1	0.4	2	0	2	1
42	45	1	0	104	208	0	0	148	1	3.0	1	0	2	1

Description

In [8]:

```
dataset.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	olc
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0(
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.0:
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.1(
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.0(
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.0(
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.8(
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.6(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.2(

In [9]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [105]:
```
###Luckily, we have no missing values
```

Let's understand our columns better:

In [10]:

```
info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic","resting blood pressure"," serum cholestoral in mg/dl","fasting blood sugar > 120 mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum heart rate achieved","exercise induced angina","oldpeak = ST depression induced by exercise relative to rest","the slope of the peak exercise ST segment","number of major vessels (0-3) colored by flourosopy","thal: 3 = normal; 6 = fixed defect; 7 = reversable defect"]



for i in range(len(info)):
    print(dataset.columns[i]+":\t\t\t"+info[i])
```

```
age:                        age
sex:                        1: male, 0: female
cp:                         chest pain type, 1: typical angina, 2: atypi
cal angina, 3: non-anginal pain, 4: asymptomatic
trestbps:                           resting blood pressure
chol:                        serum cholestoral in mg/dl
fbs:                        fasting blood sugar > 120 mg/dl
restecg:                            resting electrocardiographic results
(values 0,1,2)
thalach:                             maximum heart rate achieved
exang:                      exercise induced angina
oldpeak:                            oldpeak = ST depression induced by e
xercise relative to rest
slope:                      the slope of the peak exercise ST segment
ca:                         number of major vessels (0-3) colored by flo
urosopy
thal:                       thal: 3 = normal; 6 = fixed defect; 7 = reve
rsable defect
```

Analysing the 'target' variable

In [11]:
```
dataset["target"].describe()
```

Out[11]:
```
count    303.000000
mean       0.544554
std        0.498835
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: target, dtype: float64
```

In [108]:
```
dataset["target"].unique()
```

Out[108]:
```
array([1, 0], dtype=int64)
```

Clearly, this is a classification problem, with the target variable having values '0' and '1'

Checking correlation between columns

In [12]:
```
print(dataset.corr()["target"].abs().sort_values(ascending=False))
```

```
target      1.000000
exang       0.436757
cp          0.433798
oldpeak     0.430696
thalach     0.421741
ca          0.391724
slope       0.345877
thal        0.344029
sex         0.280937
age         0.225439
trestbps    0.144931
restecg     0.137230
chol        0.085239
fbs         0.028046
Name: target, dtype: float64
```

In [14]:
```
#This shows that most columns are moderately correlated with target, but 'fbs' is very weakly correlated.
```

In [ ]:

Exploratory Data Analysis (EDA)

First, analysing the target variable:

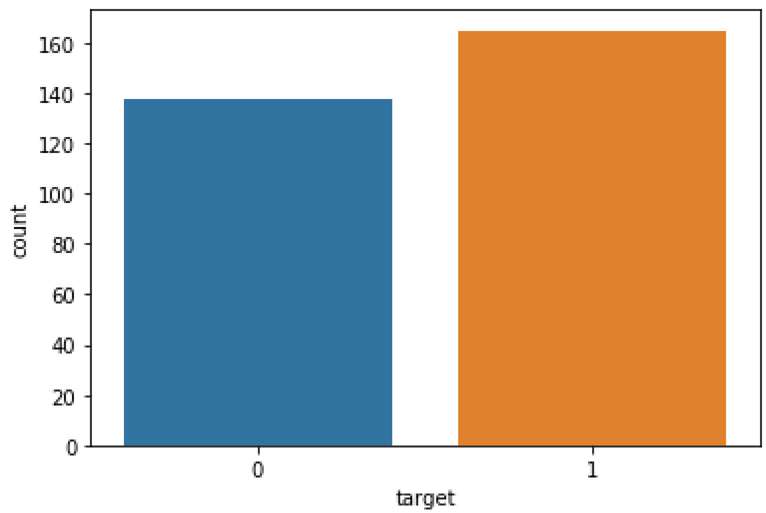
In [13]:
```
y = dataset["target"]

sns.countplot(y)


target_temp = dataset.target.value_counts()

print(target_temp)
```

```
1    165
0    138
Name: target, dtype: int64
```



In [14]:
```
print("Percentage of patience without heart problems: "+str(round(target_temp[0]*100/303,2)))
print("Percentage of patience with heart problems: "+str(round(target_temp[1]*100/303,2)))

#Alternatively,
# print("Percentage of patience with heart problems: "+str(y.where(y==1).count()*100/303))
# print("Percentage of patience with heart problems: "+str(y.where(y==0).count()*100/303))

# #Or,
# countNoDisease = len(df[df.target == 0])
# countHaveDisease = len(df[df.target == 1])
```

```
Percentage of patience without heart problems: 45.54
Percentage of patience with heart problems: 54.46
```

We'll analyse 'sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca' and 'thal' features

Analysing the 'Sex' feature

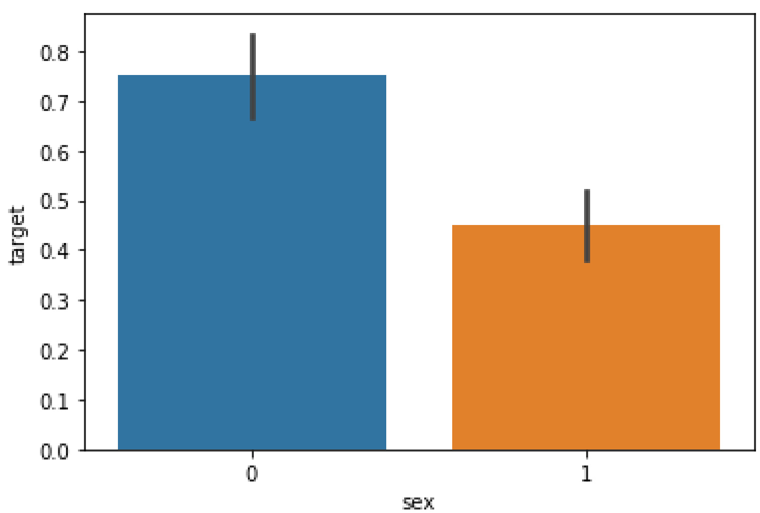
In [15]:
```
dataset["sex"].unique()
```

Out[15]:
```
array([1, 0], dtype=int64)
```

We notice, that as expected, the 'sex' feature has 2 unique features

In [16]:
```
sns.barplot(dataset["sex"],y)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164e6fd388>
```



We notice, that females are more likely to have heart problems than males

Analysing the 'Chest Pain Type' feature

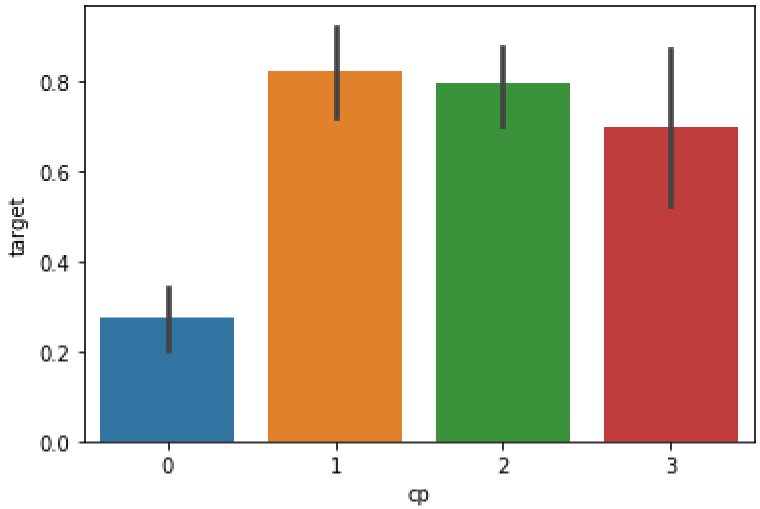
In [17]:
```
dataset["cp"].unique()
```

Out[17]:
```
array([3, 2, 1, 0], dtype=int64)
```

As expected, the CP feature has values from 0 to 3

In [18]:
```
sns.barplot(dataset["cp"],y)
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x2164e741f48>



We notice, that chest pain of '0', i.e. the ones with typical angina are much less likely to have heart problems

Analysing the FBS feature

In [19]:
```
dataset["fbs"].describe()
```

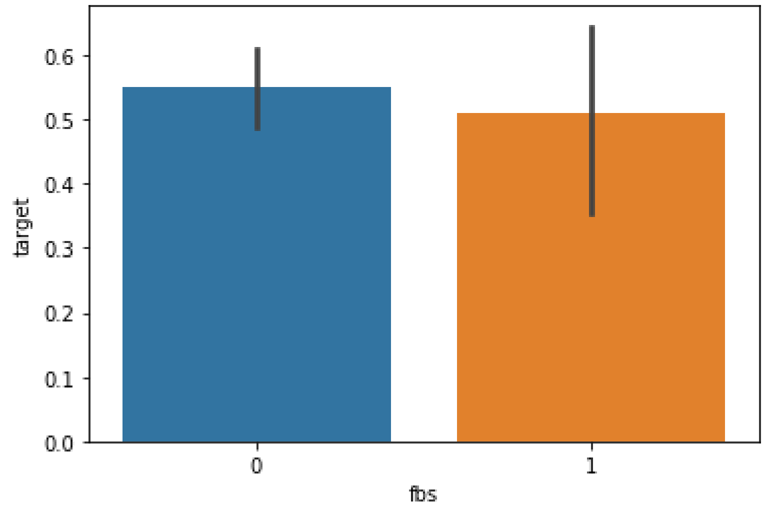
Out[19]:
```
count    303.000000
mean       0.148515
std        0.356198
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
Name: fbs, dtype: float64
```

In [20]:
```
dataset["fbs"].unique()
```

Out[20]:
```
array([1, 0], dtype=int64)
```

In [21]:
```
sns.barplot(dataset["fbs"],y)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x2164e7c0308>



Nothing extraordinary here

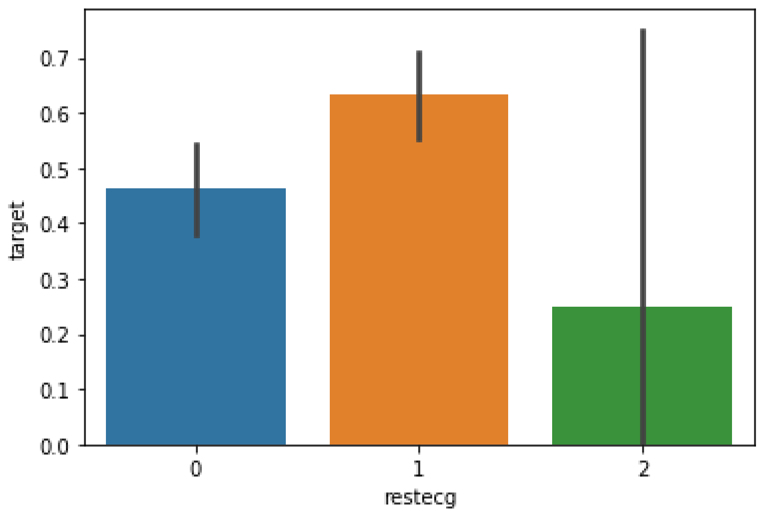
Analysing the restecg feature

In [22]:
```
dataset["restecg"].unique()
```

Out[22]: array([0, 1, 2], dtype=int64)

In [23]:
```
sns.barplot(dataset["restecg"],y)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164e74ee08>
```



We realize that people with restecg '1' and '0' are much more likely to have a heart disease than with restecg '2'

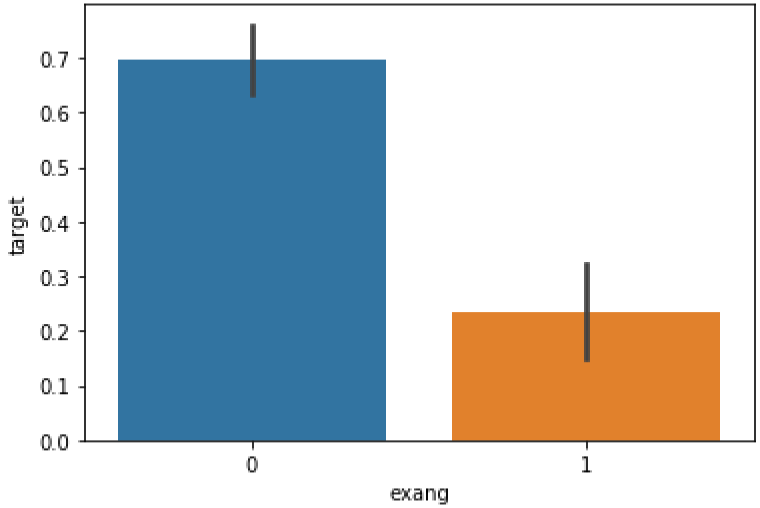
Analysing the 'exang' feature

In [24]:
```
dataset["exang"].unique()
```

Out[24]:
```
array([0, 1], dtype=int64)
```

In [25]:
```
sns.barplot(dataset["exang"],y)
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164e895708>
```



People with exang=1 i.e. Exercise induced angina are much less likely to have heart problems

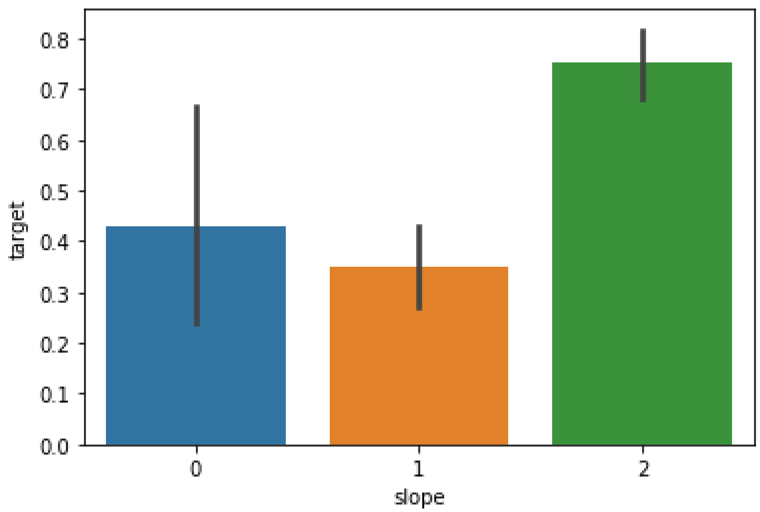
Analysing the Slope feature

In [26]:
```
dataset["slope"].unique()
```

Out[26]:
```
array([0, 2, 1], dtype=int64)
```

In [27]:
```
sns.barplot(dataset["slope"],y)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164e8fa888>
```



We observe, that Slope '2' causes heart pain much more than Slope '0' and '1'

Analysing the 'ca' feature

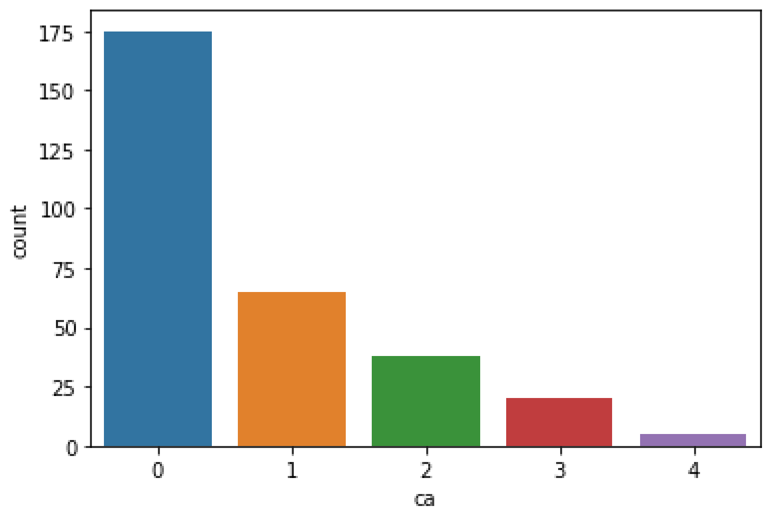
In [30]:
```
#number of major vessels (0-3) colored by flourosopy
```

In [31]:
```
dataset["ca"].unique()
```

Out[31]:
```
array([0, 2, 1, 3, 4], dtype=int64)
```

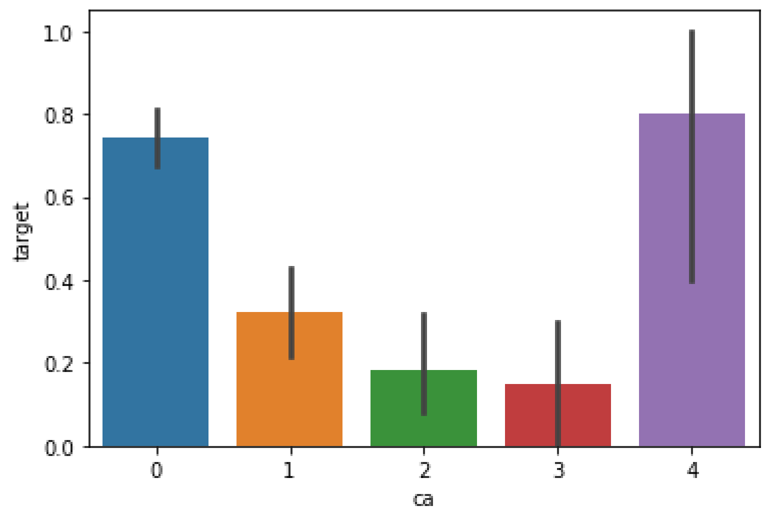
In [32]:
```
sns.countplot(dataset["ca"])
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164ea51d08>
```



In [33]:
```
sns.barplot(dataset["ca"],y)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164e654588>
```



ca=4 has astonishingly large number of heart patients

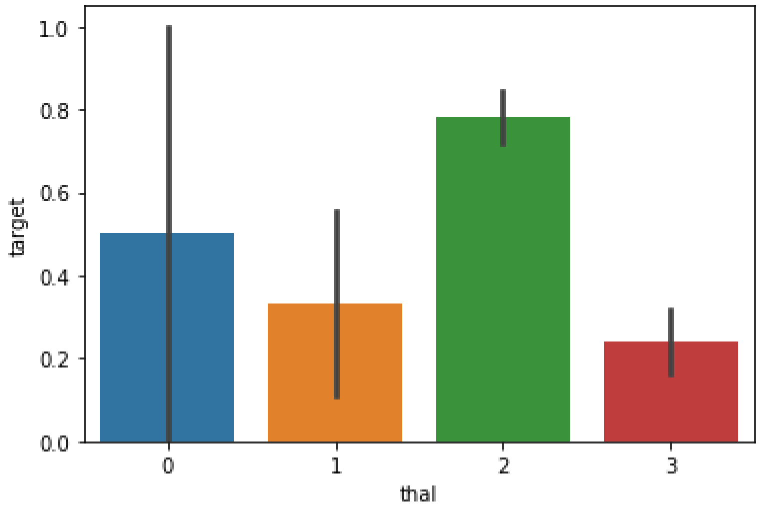
In [34]:
```
### Analysing the 'thal' feature
```

In [34]:
```
dataset["thal"].unique()
```

Out[34]:
```
array([1, 2, 3, 0], dtype=int64)
```

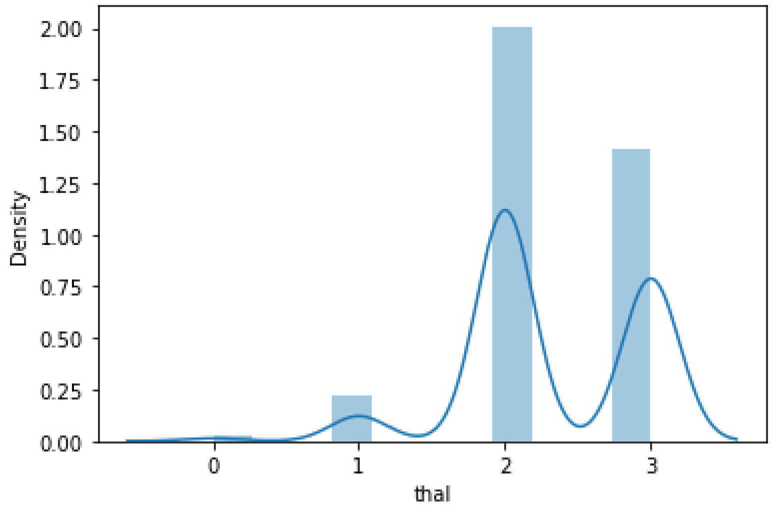
In [35]:
```
sns.barplot(dataset["thal"],y)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164ead47c8>
```



In [36]:
```
sns.distplot(dataset["thal"])
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2164eb2d2c8>
```



In [ ]:

In [77]:
```
categorical_val = []
continous_val = []
for column in dataset.columns:
    print('==============================')
    print(f"{column} : {dataset[column].unique()}")
    if len(dataset[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

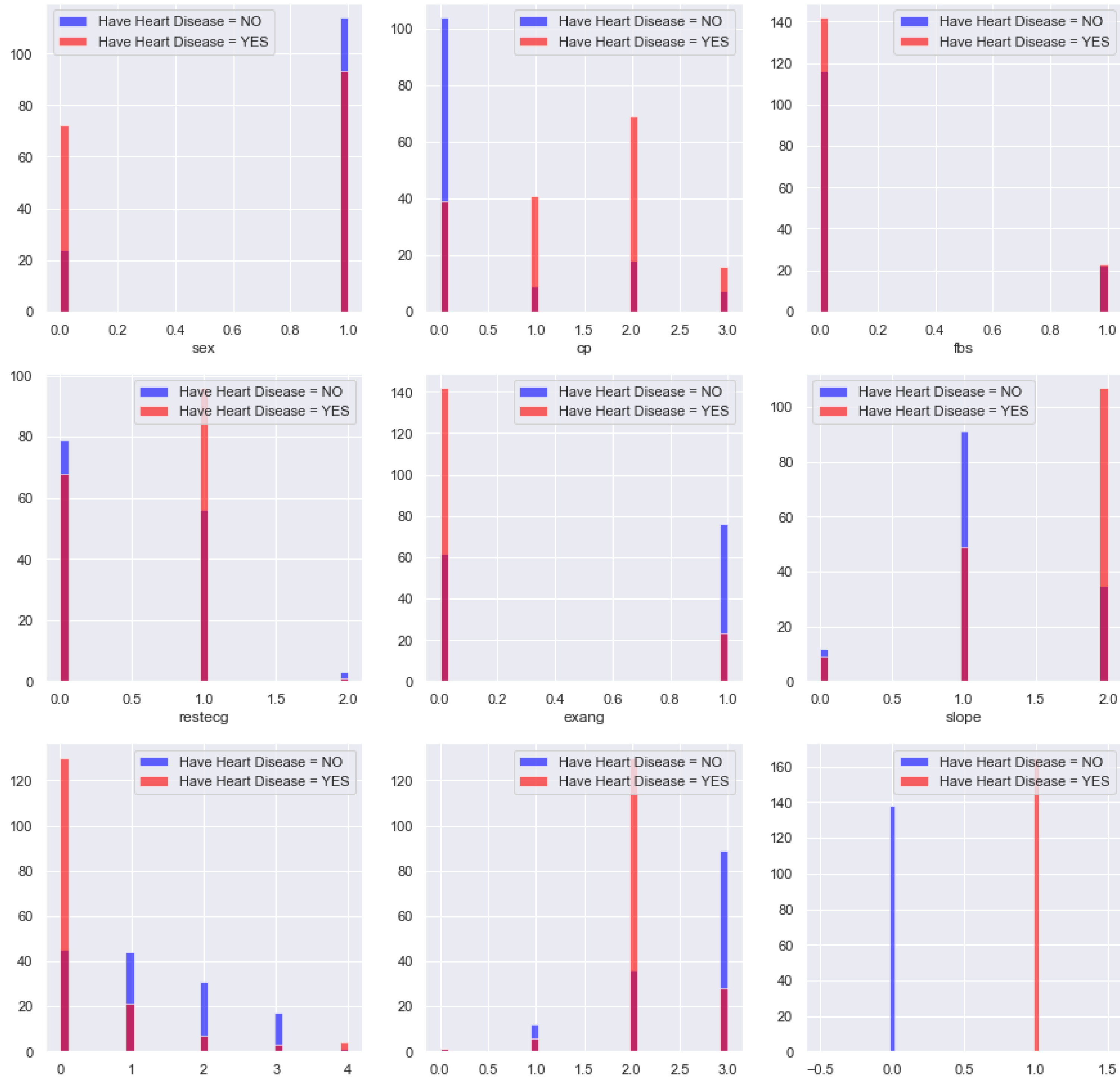
```
==============================
age : [63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 7
1 51 65 53
 46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 38 77]
==============================
sex : [1 0]
==============================
cp : [3 2 1 0]
==============================
trestbps : [145 130 120 140 172 150 110 135 160 105 125 142 155 104
138 128 108 134
 122 115 118 100 124  94 112 102 152 101 132 148 178 129 180 136 126
106
 156 170 146 117 200 165 174 192 144 123 154 114 164]
==============================
chol : [233 250 204 236 354 192 294 263 199 168 239 275 266 211 283
219 340 226
 247 234 243 302 212 175 417 197 198 177 273 213 304 232 269 360 308
245
 208 264 321 325 235 257 216 256 231 141 252 201 222 260 182 303 265
309
 186 203 183 220 209 258 227 261 221 205 240 318 298 564 277 214 248
255
 207 223 288 160 394 315 246 244 270 195 196 254 126 313 262 215 193
271
```

```
 268 267 210 295 306 178 242 180 228 149 278 253 342 157 286 229 284
224
 206 167 230 335 276 353 225 330 290 172 305 188 282 185 326 274 164
307
 249 341 407 217 174 281 289 322 299 300 293 184 409 259 200 327 237
218
 319 166 311 169 187 176 241 131]
==============================
fbs : [1 0]
==============================
restecg : [0 1 2]
==============================
thalach : [150 187 172 178 163 148 153 173 162 174 160 139 171 144 1
58 114 151 161
 179 137 157 123 152 168 140 188 125 170 165 142 180 143 182 156 115
149
 146 175 186 185 159 130 190 132 147 154 202 166 164 184 122 169 138
111
 145 194 131 133 155 167 192 121  96 126 105 181 116 108 129 120 112
128
 109 113  99 177 141 136  97 127 103 124  88 195 106  95 117  71 118
134
  90]
==============================
exang : [0 1]
```

```
===============================
oldpeak : [2.3 3.5 1.4 0.8 0.6 0.4 1.3 0.  0.5 1.6 1.2 0.2 1.8 1.
2.6 1.5 3.  2.4
 0.1 1.9 4.2 1.1 2.  0.7 0.3 0.9 3.6 3.1 3.2 2.5 2.2 2.8 3.4 6.2 4.
5.6
 2.9 2.1 3.8 4.4]
===============================
slope : [0 2 1]
===============================
ca : [0 2 1 3 4]
===============================
thal : [1 2 3 0]
===============================
target : [1 0]
```

In [78]:
```
plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    dataset[dataset["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    dataset[dataset["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```


Have Heart Disease = NO
Have Heart Disease = YES
sex
cp
fbs
restecg
exang
slope

ca thal target

IV. Train Test split

In [37]:
```
from sklearn.model_selection import train_test_split

predictors = dataset.drop("target",axis=1)
target = dataset["target"]

X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test_
size=0.20,random_state=0)
```

In [38]:
```
X_train.shape
```

Out[38]:
```
(242, 13)
```

In [39]:
```
X_test.shape
```

Out[39]:
```
(61, 13)
```

In [40]:
```
Y_train.shape
```

Out[40]:
```
(242,)
```

In [41]:
```
Y_test.shape
```

Out[41]:
```
(61,)
```

V. Model Fitting

In [43]:
```
from sklearn.metrics import accuracy_score
```

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X_train,Y_train)

Y_pred_lr = lr.predict(X_test)
```

In [45]:
```
Y_pred_lr.shape
```

Out[45]:
```
(61,)
```

In [46]:
```
score_lr = round(accuracy_score(Y_pred_lr,Y_test)*100,2)

print("The accuracy score achieved using Logistic Regression is: "+str(score_lr)+" %")
```

```
The accuracy score achieved using Logistic Regression is: 85.25 %
```

Naive Bayes

In [47]:
```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X_train,Y_train)

Y_pred_nb = nb.predict(X_test)
```

In [48]:
```
Y_pred_nb.shape
```

Out[48]:
```
(61,)
```

In [50]:
```
score_nb = round(accuracy_score(Y_pred_nb,Y_test)*100,2)

print("The accuracy score achieved using Naive Bayes is: "+str(score_nb)
+" %")
```

```
The accuracy score achieved using Naive Bayes is: 85.25 %
```

SVM

In [51]:
```
from sklearn import svm

sv = svm.SVC(kernel='linear')

sv.fit(X_train, Y_train)

Y_pred_svm = sv.predict(X_test)
```

In [52]:
```
Y_pred_svm.shape
```

Out[52]:
```
(61,)
```

In [53]:
```
score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)

print("The accuracy score achieved using Linear SVM is: "+str(score_svm)
+" %")
```

```
The accuracy score achieved using Linear SVM is: 81.97 %
```

K Nearest Neighbors

In [55]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
Y_pred_knn=knn.predict(X_test)
```

In [56]:
```
Y_pred_knn.shape
```

Out[56]:
```
(61,)
```

In [57]:
```
score_knn = round(accuracy_score(Y_pred_knn,Y_test)*100,2)

print("The accuracy score achieved using KNN is: "+str(score_knn)+" %")
```

```
The accuracy score achieved using KNN is: 67.21 %
```

Decision Tree

In [58]:
```
from sklearn.tree import DecisionTreeClassifier

max_accuracy = 0


for x in range(200):
    dt = DecisionTreeClassifier(random_state=x)
    dt.fit(X_train,Y_train)
    Y_pred_dt = dt.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

#print(max_accuracy)
#print(best_x)


dt = DecisionTreeClassifier(random_state=best_x)
dt.fit(X_train,Y_train)
Y_pred_dt = dt.predict(X_test)
```

In [59]:
```
print(Y_pred_dt.shape)
```

```
(61,)
```

In [60]:
```
score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)

print("The accuracy score achieved using Decision Tree is: "+str(score_dt)+" %")
```

```
The accuracy score achieved using Decision Tree is: 81.97 %
```

Random Forest

In [61]:
```
from sklearn.ensemble import RandomForestClassifier

max_accuracy = 0


for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train,Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

#print(max_accuracy)
#print(best_x)

rf = RandomForestClassifier(random_state=best_x)
rf.fit(X_train,Y_train)
Y_pred_rf = rf.predict(X_test)
```

In [62]:
```
Y_pred_rf.shape
```
Out[62]:
```
(61,)
```

In [63]:
```
score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)

print("The accuracy score achieved using Decision Tree is: "+str(score_r
f)+" %")
```
```
The accuracy score achieved using Decision Tree is: 90.16 %
```

XGBoost

In [65]:
```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=
42)
xgb_model.fit(X_train, Y_train)

Y_pred_xgb = xgb_model.predict(X_test)
```

In [66]:
```
Y_pred_xgb.shape
```

Out[66]:
```
(61,)
```

In [67]:
```
score_xgb = round(accuracy_score(Y_pred_xgb,Y_test)*100,2)

print("The accuracy score achieved using XGBoost is: "+str(score_xgb)+" %")
```

```
The accuracy score achieved using XGBoost is: 85.25 %
```

Neural Network

In [68]:
```
from keras.models import Sequential
from keras.layers import Dense
```

```
Using TensorFlow backend.
```

In [69]:

```
# https://stats.stackexchange.com/a/136542 helped a lot in avoiding over
fitting

model = Sequential()
model.add(Dense(11,activation='relu',input_dim=13))
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accu
racy'])
```

In [70]:
```
model.fit(X_train,Y_train,epochs=300)
```

```
Epoch 1/300
242/242 [==============================] - 2s 9ms/step - loss: 30.52
13 - accuracy: 0.4587
Epoch 2/300
242/242 [==============================] - 0s 62us/step - loss: 25.2
552 - accuracy: 0.4587
Epoch 3/300
242/242 [==============================] - 0s 62us/step - loss: 19.5
703 - accuracy: 0.4587
Epoch 4/300
242/242 [==============================] - 0s 87us/step - loss: 13.7
574 - accuracy: 0.4587
Epoch 5/300
242/242 [==============================] - 0s 91us/step - loss: 8.28
02 - accuracy: 0.4504
Epoch 6/300
242/242 [==============================] - 0s 62us/step - loss: 5.09
95 - accuracy: 0.4669
Epoch 7/300
242/242 [==============================] - 0s 78us/step - loss: 4.54
52 - accuracy: 0.5455
Epoch 8/300
242/242 [==============================] - 0s 107us/step - loss: 4.6
416 - accuracy: 0.5372
Epoch 9/300
```

```
242/242 [==============================] - 0s 70us/step - loss: 4.37
92 - accuracy: 0.5537
Epoch 10/300
242/242 [==============================] - 0s 58us/step - loss: 4.19
85 - accuracy: 0.5620
Epoch 11/300
242/242 [==============================] - 0s 74us/step - loss: 4.04
59 - accuracy: 0.5455
Epoch 12/300
242/242 [==============================] - 0s 82us/step - loss: 3.81
98 - accuracy: 0.5496
Epoch 13/300
242/242 [==============================] - 0s 62us/step - loss: 3.44
86 - accuracy: 0.5620
Epoch 14/300
242/242 [==============================] - 0s 74us/step - loss: 3.03
52 - accuracy: 0.5661
Epoch 15/300
242/242 [==============================] - 0s 66us/step - loss: 2.77
66 - accuracy: 0.5496
Epoch 16/300
242/242 [==============================] - 0s 74us/step - loss: 2.54
85 - accuracy: 0.5455
Epoch 17/300
242/242 [==============================] - 0s 86us/step - loss: 2.37
```

```
07 - accuracy: 0.5702
Epoch 18/300
242/242 [==============================] - 0s 82us/step - loss: 2.19
00 - accuracy: 0.5744
Epoch 19/300
242/242 [==============================] - 0s 74us/step - loss: 1.99
69 - accuracy: 0.5537
Epoch 20/300
242/242 [==============================] - 0s 70us/step - loss: 1.83
45 - accuracy: 0.5496
Epoch 21/300
242/242 [==============================] - 0s 82us/step - loss: 1.71
05 - accuracy: 0.5537
Epoch 22/300
242/242 [==============================] - 0s 74us/step - loss: 1.55
83 - accuracy: 0.5537
Epoch 23/300
242/242 [==============================] - 0s 66us/step - loss: 1.44
31 - accuracy: 0.5496
Epoch 24/300
242/242 [==============================] - 0s 78us/step - loss: 1.35
18 - accuracy: 0.5620
Epoch 25/300
242/242 [==============================] - 0s 91us/step - loss: 1.23
31 - accuracy: 0.5702
```

```
Epoch 26/300
242/242 [==============================] - 0s 70us/step - loss: 1.17
85 - accuracy: 0.5744
Epoch 27/300
242/242 [==============================] - 0s 74us/step - loss: 1.09
51 - accuracy: 0.5537
Epoch 28/300
242/242 [==============================] - 0s 91us/step - loss: 1.04
13 - accuracy: 0.5620
Epoch 29/300
242/242 [==============================] - 0s 95us/step - loss: 0.99
62 - accuracy: 0.5537
Epoch 30/300
242/242 [==============================] - 0s 78us/step - loss: 0.97
01 - accuracy: 0.5702
Epoch 31/300
242/242 [==============================] - 0s 70us/step - loss: 0.95
00 - accuracy: 0.5950
Epoch 32/300
242/242 [==============================] - 0s 82us/step - loss: 0.89
97 - accuracy: 0.5868
Epoch 33/300
242/242 [==============================] - 0s 87us/step - loss: 0.89
02 - accuracy: 0.6033
Epoch 34/300
```

```
242/242 [==============================] - 0s 74us/step - loss: 0.88
73 - accuracy: 0.6281
Epoch 35/300
242/242 [==============================] - 0s 87us/step - loss: 0.91
38 - accuracy: 0.5620
Epoch 36/300
242/242 [==============================] - 0s 66us/step - loss: 0.85
27 - accuracy: 0.6240
Epoch 37/300
242/242 [==============================] - 0s 78us/step - loss: 0.82
98 - accuracy: 0.5868
Epoch 38/300
242/242 [==============================] - 0s 54us/step - loss: 0.79
17 - accuracy: 0.6446
Epoch 39/300
242/242 [==============================] - 0s 99us/step - loss: 0.82
60 - accuracy: 0.6570
Epoch 40/300
242/242 [==============================] - 0s 62us/step - loss: 0.78
04 - accuracy: 0.5950
Epoch 41/300
242/242 [==============================] - 0s 70us/step - loss: 0.79
50 - accuracy: 0.6570
Epoch 42/300
242/242 [==============================] - 0s 62us/step - loss: 0.74
```

```
66 - accuracy: 0.6364
Epoch 43/300
242/242 [==============================] - 0s 62us/step - loss: 0.73
12 - accuracy: 0.6240
Epoch 44/300
242/242 [==============================] - 0s 62us/step - loss: 0.71
24 - accuracy: 0.6612
Epoch 45/300
242/242 [==============================] - 0s 62us/step - loss: 0.71
62 - accuracy: 0.6405
Epoch 46/300
242/242 [==============================] - 0s 54us/step - loss: 0.70
19 - accuracy: 0.6446
Epoch 47/300
242/242 [==============================] - 0s 62us/step - loss: 0.68
78 - accuracy: 0.6860
Epoch 48/300
242/242 [==============================] - 0s 66us/step - loss: 0.66
62 - accuracy: 0.6736
Epoch 49/300
242/242 [==============================] - 0s 58us/step - loss: 0.65
78 - accuracy: 0.6736
Epoch 50/300
242/242 [==============================] - 0s 82us/step - loss: 0.65
63 - accuracy: 0.6612
```

```
Epoch 51/300
242/242 [==============================] - 0s 62us/step - loss: 0.64
67 - accuracy: 0.6570
Epoch 52/300
242/242 [==============================] - 0s 74us/step - loss: 0.64
21 - accuracy: 0.6942
Epoch 53/300
242/242 [==============================] - 0s 62us/step - loss: 0.62
41 - accuracy: 0.6983
Epoch 54/300
242/242 [==============================] - 0s 74us/step - loss: 0.60
92 - accuracy: 0.7066
Epoch 55/300
242/242 [==============================] - 0s 54us/step - loss: 0.59
75 - accuracy: 0.6942
Epoch 56/300
242/242 [==============================] - 0s 95us/step - loss: 0.61
48 - accuracy: 0.6860
Epoch 57/300
242/242 [==============================] - 0s 74us/step - loss: 0.61
32 - accuracy: 0.7190
Epoch 58/300
242/242 [==============================] - 0s 62us/step - loss: 0.58
89 - accuracy: 0.6818
Epoch 59/300
```

```
242/242 [==============================] - 0s 66us/step - loss: 0.59
12 - accuracy: 0.7314
Epoch 60/300
242/242 [==============================] - 0s 62us/step - loss: 0.58
49 - accuracy: 0.7066
Epoch 61/300
242/242 [==============================] - 0s 58us/step - loss: 0.55
10 - accuracy: 0.7231
Epoch 62/300
242/242 [==============================] - 0s 49us/step - loss: 0.55
25 - accuracy: 0.7190
Epoch 63/300
242/242 [==============================] - 0s 58us/step - loss: 0.55
55 - accuracy: 0.6983
Epoch 64/300
242/242 [==============================] - 0s 58us/step - loss: 0.53
50 - accuracy: 0.7397
Epoch 65/300
242/242 [==============================] - 0s 58us/step - loss: 0.51
35 - accuracy: 0.7273
Epoch 66/300
242/242 [==============================] - 0s 62us/step - loss: 0.50
64 - accuracy: 0.7521
Epoch 67/300
242/242 [==============================] - 0s 66us/step - loss: 0.50
```

```
92 - accuracy: 0.7355
Epoch 68/300
242/242 [==============================] - 0s 54us/step - loss: 0.50
37 - accuracy: 0.7645
Epoch 69/300
242/242 [==============================] - 0s 54us/step - loss: 0.50
91 - accuracy: 0.7355
Epoch 70/300
242/242 [==============================] - 0s 54us/step - loss: 0.51
77 - accuracy: 0.7645
Epoch 71/300
242/242 [==============================] - 0s 78us/step - loss: 0.50
18 - accuracy: 0.7603
Epoch 72/300
242/242 [==============================] - 0s 91us/step - loss: 0.47
65 - accuracy: 0.7603
Epoch 73/300
242/242 [==============================] - 0s 82us/step - loss: 0.48
50 - accuracy: 0.7521
Epoch 74/300
242/242 [==============================] - 0s 66us/step - loss: 0.48
08 - accuracy: 0.7686
Epoch 75/300
242/242 [==============================] - 0s 66us/step - loss: 0.46
74 - accuracy: 0.7645
```

```
Epoch 76/300
242/242 [==============================] - 0s 78us/step - loss: 0.46
50 - accuracy: 0.7686
Epoch 77/300
242/242 [==============================] - 0s 78us/step - loss: 0.46
60 - accuracy: 0.7769
Epoch 78/300
242/242 [==============================] - 0s 78us/step - loss: 0.45
02 - accuracy: 0.7769
Epoch 79/300
242/242 [==============================] - 0s 82us/step - loss: 0.44
87 - accuracy: 0.7975
Epoch 80/300
242/242 [==============================] - 0s 70us/step - loss: 0.44
47 - accuracy: 0.7769
Epoch 81/300
242/242 [==============================] - 0s 74us/step - loss: 0.43
91 - accuracy: 0.7851
Epoch 82/300
242/242 [==============================] - 0s 58us/step - loss: 0.44
22 - accuracy: 0.7893
Epoch 83/300
242/242 [==============================] - 0s 78us/step - loss: 0.43
45 - accuracy: 0.7893
Epoch 84/300
```

```
242/242 [==============================] - 0s 74us/step - loss: 0.43
39 - accuracy: 0.7810
Epoch 85/300
242/242 [==============================] - 0s 87us/step - loss: 0.42
82 - accuracy: 0.7893
Epoch 86/300
242/242 [==============================] - 0s 70us/step - loss: 0.42
26 - accuracy: 0.7934
Epoch 87/300
242/242 [==============================] - 0s 70us/step - loss: 0.42
55 - accuracy: 0.8058
Epoch 88/300
242/242 [==============================] - 0s 74us/step - loss: 0.44
17 - accuracy: 0.7934
Epoch 89/300
242/242 [==============================] - 0s 66us/step - loss: 0.42
13 - accuracy: 0.8140
Epoch 90/300
242/242 [==============================] - 0s 66us/step - loss: 0.41
54 - accuracy: 0.7975
Epoch 91/300
242/242 [==============================] - 0s 78us/step - loss: 0.40
90 - accuracy: 0.8058
Epoch 92/300
242/242 [==============================] - 0s 70us/step - loss: 0.42
```

```
32 - accuracy: 0.7975
Epoch 93/300
242/242 [==============================] - 0s 66us/step - loss: 0.43
03 - accuracy: 0.7851
Epoch 94/300
242/242 [==============================] - 0s 74us/step - loss: 0.40
67 - accuracy: 0.8182
Epoch 95/300
242/242 [==============================] - 0s 70us/step - loss: 0.40
36 - accuracy: 0.8099
Epoch 96/300
242/242 [==============================] - 0s 66us/step - loss: 0.39
95 - accuracy: 0.8099
Epoch 97/300
242/242 [==============================] - 0s 70us/step - loss: 0.39
88 - accuracy: 0.8223
Epoch 98/300
242/242 [==============================] - 0s 70us/step - loss: 0.40
30 - accuracy: 0.8223
Epoch 99/300
242/242 [==============================] - 0s 74us/step - loss: 0.40
39 - accuracy: 0.8223
Epoch 100/300
242/242 [==============================] - 0s 66us/step - loss: 0.41
79 - accuracy: 0.7975
```

```
Epoch 101/300
242/242 [==============================] - 0s 66us/step - loss: 0.39
50 - accuracy: 0.8140
Epoch 102/300
242/242 [==============================] - 0s 58us/step - loss: 0.40
17 - accuracy: 0.8223
Epoch 103/300
242/242 [==============================] - 0s 58us/step - loss: 0.41
61 - accuracy: 0.8264
Epoch 104/300
242/242 [==============================] - 0s 54us/step - loss: 0.43
52 - accuracy: 0.7975
Epoch 105/300
242/242 [==============================] - 0s 66us/step - loss: 0.42
40 - accuracy: 0.8058
Epoch 106/300
242/242 [==============================] - 0s 66us/step - loss: 0.38
94 - accuracy: 0.8347
Epoch 107/300
242/242 [==============================] - 0s 66us/step - loss: 0.39
33 - accuracy: 0.8264
Epoch 108/300
242/242 [==============================] - 0s 62us/step - loss: 0.38
75 - accuracy: 0.8264
Epoch 109/300
```

```
242/242 [==============================] - 0s 70us/step - loss: 0.38
33 - accuracy: 0.8347
Epoch 110/300
242/242 [==============================] - 0s 66us/step - loss: 0.38
92 - accuracy: 0.8264
Epoch 111/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
24 - accuracy: 0.8306
Epoch 112/300
242/242 [==============================] - 0s 70us/step - loss: 0.39
27 - accuracy: 0.8306
Epoch 113/300
242/242 [==============================] - 0s 78us/step - loss: 0.40
25 - accuracy: 0.8099
Epoch 114/300
242/242 [==============================] - 0s 58us/step - loss: 0.39
39 - accuracy: 0.8099
Epoch 115/300
242/242 [==============================] - 0s 74us/step - loss: 0.38
01 - accuracy: 0.8430
Epoch 116/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
92 - accuracy: 0.8430
Epoch 117/300
242/242 [==============================] - 0s 58us/step - loss: 0.38
```

```
54 - accuracy: 0.8347
Epoch 118/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
90 - accuracy: 0.8306
Epoch 119/300
242/242 [==============================] - 0s 74us/step - loss: 0.38
51 - accuracy: 0.8347
Epoch 120/300
242/242 [==============================] - 0s 62us/step - loss: 0.37
62 - accuracy: 0.8554
Epoch 121/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
92 - accuracy: 0.8512
Epoch 122/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
91 - accuracy: 0.8264
Epoch 123/300
242/242 [==============================] - 0s 54us/step - loss: 0.37
54 - accuracy: 0.8471
Epoch 124/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
54 - accuracy: 0.8512
Epoch 125/300
242/242 [==============================] - 0s 66us/step - loss: 0.39
10 - accuracy: 0.8264
```

```
Epoch 126/300
242/242 [==============================] - 0s 70us/step - loss: 0.39
43 - accuracy: 0.8017
Epoch 127/300
242/242 [==============================] - 0s 82us/step - loss: 0.39
25 - accuracy: 0.8306
Epoch 128/300
242/242 [==============================] - 0s 58us/step - loss: 0.39
04 - accuracy: 0.8182
Epoch 129/300
242/242 [==============================] - ETA: 0s - loss: 0.3297 -
accuracy: 0.84 - 0s 66us/step - loss: 0.4215 - accuracy: 0.7934
Epoch 130/300
242/242 [==============================] - 0s 66us/step - loss: 0.39
20 - accuracy: 0.8430
Epoch 131/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
75 - accuracy: 0.8306
Epoch 132/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
38 - accuracy: 0.8512
Epoch 133/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
36 - accuracy: 0.8512
Epoch 134/300
```

```
242/242 [==============================] - 0s 62us/step - loss: 0.37
50 - accuracy: 0.8388
Epoch 135/300
242/242 [==============================] - 0s 58us/step - loss: 0.40
47 - accuracy: 0.8306
Epoch 136/300
242/242 [==============================] - 0s 58us/step - loss: 0.39
85 - accuracy: 0.8140
Epoch 137/300
242/242 [==============================] - 0s 74us/step - loss: 0.37
38 - accuracy: 0.8512
Epoch 138/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
27 - accuracy: 0.8388
Epoch 139/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
20 - accuracy: 0.8512
Epoch 140/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
70 - accuracy: 0.8223
Epoch 141/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
54 - accuracy: 0.8430
Epoch 142/300
242/242 [==============================] - 0s 54us/step - loss: 0.38
```

```
93 - accuracy: 0.8347
Epoch 143/300
242/242 [==============================] - 0s 66us/step - loss: 0.38
83 - accuracy: 0.8223
Epoch 144/300
242/242 [==============================] - 0s 62us/step - loss: 0.37
98 - accuracy: 0.8306
Epoch 145/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
06 - accuracy: 0.8595
Epoch 146/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
90 - accuracy: 0.8471
Epoch 147/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
37 - accuracy: 0.8388
Epoch 148/300
242/242 [==============================] - 0s 66us/step - loss: 0.38
14 - accuracy: 0.8347
Epoch 149/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
71 - accuracy: 0.8264
Epoch 150/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
53 - accuracy: 0.8595
```

```
Epoch 151/300
242/242 [==============================] - 0s 58us/step - loss: 0.38
90 - accuracy: 0.8306
Epoch 152/300
242/242 [==============================] - 0s 66us/step - loss: 0.40
60 - accuracy: 0.8140
Epoch 153/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
26 - accuracy: 0.8306
Epoch 154/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
62 - accuracy: 0.8430
Epoch 155/300
242/242 [==============================] - 0s 66us/step - loss: 0.38
48 - accuracy: 0.8306
Epoch 156/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
59 - accuracy: 0.8264
Epoch 157/300
242/242 [==============================] - 0s 62us/step - loss: 0.39
70 - accuracy: 0.8182
Epoch 158/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
33 - accuracy: 0.8347
Epoch 159/300
```

```
242/242 [==============================] - 0s 49us/step - loss: 0.36
79 - accuracy: 0.8430
Epoch 160/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
33 - accuracy: 0.8554
Epoch 161/300
242/242 [==============================] - 0s 58us/step - loss: 0.36
51 - accuracy: 0.8554
Epoch 162/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
17 - accuracy: 0.8430
Epoch 163/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
82 - accuracy: 0.8306
Epoch 164/300
242/242 [==============================] - ETA: 0s - loss: 0.4591 -
accuracy: 0.78 - 0s 70us/step - loss: 0.3684 - accuracy: 0.8347
Epoch 165/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
68 - accuracy: 0.8471
Epoch 166/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
49 - accuracy: 0.8554
Epoch 167/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
```

```
07 - accuracy: 0.8554
Epoch 168/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
63 - accuracy: 0.8554
Epoch 169/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
37 - accuracy: 0.8223
Epoch 170/300
242/242 [==============================] - 0s 66us/step - loss: 0.40
90 - accuracy: 0.8347
Epoch 171/300
242/242 [==============================] - 0s 62us/step - loss: 0.38
53 - accuracy: 0.8182
Epoch 172/300
242/242 [==============================] - 0s 74us/step - loss: 0.39
04 - accuracy: 0.8140
Epoch 173/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
20 - accuracy: 0.8595
Epoch 174/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
22 - accuracy: 0.8471
Epoch 175/300
242/242 [==============================] - 0s 66us/step - loss: 0.38
42 - accuracy: 0.8017
```

```
Epoch 176/300
242/242 [==============================] - 0s 78us/step - loss: 0.39
06 - accuracy: 0.8306
Epoch 177/300
242/242 [==============================] - 0s 58us/step - loss: 0.36
74 - accuracy: 0.8554
Epoch 178/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
70 - accuracy: 0.8223
Epoch 179/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
85 - accuracy: 0.8471
Epoch 180/300
242/242 [==============================] - 0s 74us/step - loss: 0.37
86 - accuracy: 0.8223
Epoch 181/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
97 - accuracy: 0.8388
Epoch 182/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
63 - accuracy: 0.8471
Epoch 183/300
242/242 [==============================] - 0s 78us/step - loss: 0.35
61 - accuracy: 0.8471
Epoch 184/300
```

```
242/242 [==============================] - 0s 70us/step - loss: 0.36
81 - accuracy: 0.8182
Epoch 185/300
242/242 [==============================] - 0s 70us/step - loss: 0.41
48 - accuracy: 0.8140
Epoch 186/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
39 - accuracy: 0.8554
Epoch 187/300
242/242 [==============================] - 0s 62us/step - loss: 0.39
91 - accuracy: 0.8264
Epoch 188/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
04 - accuracy: 0.8430
Epoch 189/300
242/242 [==============================] - 0s 54us/step - loss: 0.36
54 - accuracy: 0.8430
Epoch 190/300
242/242 [==============================] - 0s 74us/step - loss: 0.36
35 - accuracy: 0.8554
Epoch 191/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
56 - accuracy: 0.8554
Epoch 192/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
```

```
47 - accuracy: 0.8471
Epoch 193/300
242/242 [==============================] - 0s 70us/step - loss: 0.39
45 - accuracy: 0.8058
Epoch 194/300
242/242 [==============================] - 0s 74us/step - loss: 0.38
50 - accuracy: 0.8471
Epoch 195/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
54 - accuracy: 0.8471
Epoch 196/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
48 - accuracy: 0.8512
Epoch 197/300
242/242 [==============================] - 0s 58us/step - loss: 0.36
35 - accuracy: 0.8554
Epoch 198/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
88 - accuracy: 0.8512
Epoch 199/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
62 - accuracy: 0.8264
Epoch 200/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
79 - accuracy: 0.8306
```

```
Epoch 201/300
242/242 [==============================] - 0s 74us/step - loss: 0.35
98 - accuracy: 0.8430
Epoch 202/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
85 - accuracy: 0.8512
Epoch 203/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
96 - accuracy: 0.8388
Epoch 204/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
74 - accuracy: 0.8554
Epoch 205/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
01 - accuracy: 0.8430
Epoch 206/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
97 - accuracy: 0.8471
Epoch 207/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
95 - accuracy: 0.8512
Epoch 208/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
25 - accuracy: 0.8430
Epoch 209/300
```

```
242/242 [==============================] - 0s 66us/step - loss: 0.36
23 - accuracy: 0.8512
Epoch 210/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
90 - accuracy: 0.8430
Epoch 211/300
242/242 [==============================] - 0s 62us/step - loss: 0.38
56 - accuracy: 0.8264
Epoch 212/300
242/242 [==============================] - 0s 70us/step - loss: 0.39
91 - accuracy: 0.8099
Epoch 213/300
242/242 [==============================] - 0s 62us/step - loss: 0.37
53 - accuracy: 0.8471
Epoch 214/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
57 - accuracy: 0.8636
Epoch 215/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
04 - accuracy: 0.8595
Epoch 216/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
69 - accuracy: 0.8595
Epoch 217/300
242/242 [==============================] - 0s 78us/step - loss: 0.37
```

```
99 - accuracy: 0.8306
Epoch 218/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
84 - accuracy: 0.8554
Epoch 219/300
242/242 [==============================] - 0s 58us/step - loss: 0.39
53 - accuracy: 0.8140
Epoch 220/300
242/242 [==============================] - 0s 58us/step - loss: 0.37
17 - accuracy: 0.8347
Epoch 221/300
242/242 [==============================] - 0s 78us/step - loss: 0.36
04 - accuracy: 0.8512
Epoch 222/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
64 - accuracy: 0.8306
Epoch 223/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
55 - accuracy: 0.8306
Epoch 224/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
66 - accuracy: 0.8678
Epoch 225/300
242/242 [==============================] - 0s 74us/step - loss: 0.35
73 - accuracy: 0.8306
```

```
Epoch 226/300
242/242 [==============================] - 0s 74us/step - loss: 0.35
72 - accuracy: 0.8554
Epoch 227/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
58 - accuracy: 0.8554
Epoch 228/300
242/242 [==============================] - 0s 58us/step - loss: 0.36
34 - accuracy: 0.8306
Epoch 229/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
71 - accuracy: 0.8430
Epoch 230/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
62 - accuracy: 0.8595
Epoch 231/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
84 - accuracy: 0.8554
Epoch 232/300
242/242 [==============================] - 0s 66us/step - loss: 0.38
38 - accuracy: 0.8388
Epoch 233/300
242/242 [==============================] - 0s 74us/step - loss: 0.36
10 - accuracy: 0.8512
Epoch 234/300
```

```
242/242 [==============================] - 0s 87us/step - loss: 0.36
87 - accuracy: 0.8347
Epoch 235/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
90 - accuracy: 0.8471
Epoch 236/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
95 - accuracy: 0.8595
Epoch 237/300
242/242 [==============================] - 0s 54us/step - loss: 0.36
60 - accuracy: 0.8347
Epoch 238/300
242/242 [==============================] - 0s 54us/step - loss: 0.36
50 - accuracy: 0.8347
Epoch 239/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
62 - accuracy: 0.8306
Epoch 240/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
46 - accuracy: 0.8512
Epoch 241/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
66 - accuracy: 0.8595
Epoch 242/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
```

```
22 - accuracy: 0.8595
Epoch 243/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
54 - accuracy: 0.8512
Epoch 244/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
51 - accuracy: 0.8430
Epoch 245/300
242/242 [==============================] - 0s 74us/step - loss: 0.37
15 - accuracy: 0.8223
Epoch 246/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
70 - accuracy: 0.8471
Epoch 247/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
44 - accuracy: 0.8430
Epoch 248/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
81 - accuracy: 0.8388
Epoch 249/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
65 - accuracy: 0.8471
Epoch 250/300
242/242 [==============================] - 0s 78us/step - loss: 0.37
20 - accuracy: 0.8512
```

```
Epoch 251/300
242/242 [==============================] - 0s 70us/step - loss: 0.38
41 - accuracy: 0.8223
Epoch 252/300
242/242 [==============================] - 0s 70us/step - loss: 0.39
72 - accuracy: 0.8223
Epoch 253/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
45 - accuracy: 0.8512
Epoch 254/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
45 - accuracy: 0.8471
Epoch 255/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
66 - accuracy: 0.8471
Epoch 256/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
62 - accuracy: 0.8347
Epoch 257/300
242/242 [==============================] - 0s 74us/step - loss: 0.36
97 - accuracy: 0.8471
Epoch 258/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
79 - accuracy: 0.8554
Epoch 259/300
```

```
242/242 [==============================] - 0s 70us/step - loss: 0.37
30 - accuracy: 0.8182
Epoch 260/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
74 - accuracy: 0.8512
Epoch 261/300
242/242 [==============================] - 0s 74us/step - loss: 0.37
20 - accuracy: 0.8471
Epoch 262/300
242/242 [==============================] - 0s 74us/step - loss: 0.40
96 - accuracy: 0.8099
Epoch 263/300
242/242 [==============================] - 0s 74us/step - loss: 0.37
05 - accuracy: 0.8223
Epoch 264/300
242/242 [==============================] - 0s 62us/step - loss: 0.37
55 - accuracy: 0.8554
Epoch 265/300
242/242 [==============================] - 0s 70us/step - loss: 0.34
93 - accuracy: 0.8388
Epoch 266/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
04 - accuracy: 0.8636
Epoch 267/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
```

```
41 - accuracy: 0.8430
Epoch 268/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
70 - accuracy: 0.8471
Epoch 269/300
242/242 [==============================] - 0s 74us/step - loss: 0.35
88 - accuracy: 0.8719
Epoch 270/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
53 - accuracy: 0.8512
Epoch 271/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
81 - accuracy: 0.8347
Epoch 272/300
242/242 [==============================] - 0s 70us/step - loss: 0.37
75 - accuracy: 0.8430
Epoch 273/300
242/242 [==============================] - 0s 66us/step - loss: 0.36
14 - accuracy: 0.8554
Epoch 274/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
86 - accuracy: 0.8554
Epoch 275/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
05 - accuracy: 0.8223
```

```
Epoch 276/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
97 - accuracy: 0.8471
Epoch 277/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
84 - accuracy: 0.8554
Epoch 278/300
242/242 [==============================] - 0s 54us/step - loss: 0.35
14 - accuracy: 0.8471
Epoch 279/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
12 - accuracy: 0.8595
Epoch 280/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
80 - accuracy: 0.8430
Epoch 281/300
242/242 [==============================] - 0s 74us/step - loss: 0.36
18 - accuracy: 0.8471
Epoch 282/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
49 - accuracy: 0.8430
Epoch 283/300
242/242 [==============================] - 0s 62us/step - loss: 0.35
72 - accuracy: 0.8512
Epoch 284/300
```

```
242/242 [==============================] - 0s 78us/step - loss: 0.35
34 - accuracy: 0.8430
Epoch 285/300
242/242 [==============================] - 0s 54us/step - loss: 0.35
09 - accuracy: 0.8512
Epoch 286/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
66 - accuracy: 0.8430
Epoch 287/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
38 - accuracy: 0.8430
Epoch 288/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
39 - accuracy: 0.8512
Epoch 289/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
64 - accuracy: 0.8430
Epoch 290/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
43 - accuracy: 0.8512
Epoch 291/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
68 - accuracy: 0.8388
Epoch 292/300
242/242 [==============================] - 0s 58us/step - loss: 0.35
```

```
89 - accuracy: 0.8512
Epoch 293/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
38 - accuracy: 0.8388
Epoch 294/300
242/242 [==============================] - 0s 62us/step - loss: 0.36
81 - accuracy: 0.8471
Epoch 295/300
242/242 [==============================] - 0s 70us/step - loss: 0.39
05 - accuracy: 0.8140
Epoch 296/300
242/242 [==============================] - 0s 70us/step - loss: 0.36
43 - accuracy: 0.8347
Epoch 297/300
242/242 [==============================] - 0s 66us/step - loss: 0.35
61 - accuracy: 0.8306
Epoch 298/300
242/242 [==============================] - 0s 70us/step - loss: 0.35
77 - accuracy: 0.8471
Epoch 299/300
242/242 [==============================] - 0s 66us/step - loss: 0.37
03 - accuracy: 0.8264
Epoch 300/300
242/242 [==============================] - 0s 54us/step - loss: 0.35
51 - accuracy: 0.8347
```

Out[70]: <keras.callbacks.callbacks.History at 0x216566454c8>

In [71]:
```
Y_pred_nn = model.predict(X_test)
```

In [72]:
```
Y_pred_nn.shape
```

Out[72]: (61, 1)

In [73]:
```
rounded = [round(x[0]) for x in Y_pred_nn]

Y_pred_nn = rounded
```

In [74]:
```
score_nn = round(accuracy_score(Y_pred_nn,Y_test)*100,2)

print("The accuracy score achieved using Neural Network is: "+str(score_nn)+" %")

#Note: Accuracy of 85% can be achieved on the test set, by setting epochs=2000, and number of nodes = 11.
```

```
The accuracy score achieved using Neural Network is: 81.97 %
```

VI. Output final score

In [75]:

```
scores = [score_lr,score_nb,score_svm,score_knn,score_dt,score_rf,score_xgb,score_nn]
algorithms = ["Logistic Regression","Naive Bayes","Support Vector Machine","K-Nearest Neighbors","Decision Tree","Random Forest","XGBoost","Neural Network"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Logistic Regression is: 85.25 %
The accuracy score achieved using Naive Bayes is: 85.25 %
The accuracy score achieved using Support Vector Machine is: 81.97 %
The accuracy score achieved using K-Nearest Neighbors is: 67.21 %
The accuracy score achieved using Decision Tree is: 81.97 %
The accuracy score achieved using Random Forest is: 90.16 %
The accuracy score achieved using XGBoost is: 85.25 %
The accuracy score achieved using Neural Network is: 81.97 %
```

In [76]:
```
sns.set(rc={'figure.figsize':(15,8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

Out[76]: <matplotlib.axes._subplots.AxesSubplot at 0x216569cdc48>