Text Classification

Motalib Rahim (mxr170012)

Dr. Karen Mazidi


```
import pandas as pd # for handling data
from sklearn.feature_extraction.text import TfidfVectorizer # raw text to matrix
from sklearn.model_selection import train_test_split # train test
import seaborn as sns # graph
from sklearn.naive_bayes import MultinomialNB # probabilistic method
from sklearn.naive_bayes import GaussianNB  # # probabilistic learning
import math # math functions
from sklearn import metrics
from sklearn.metrics import classification_report # evaluation performance
import numpy as np # math operation
import tensorflow as tf # train
from tensorflow import keras # models
```

```
df = pd.read_csv ('heart.csv')
```

```
df.head() # top of dataset
```

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

```
df.info() # info of dataset
```

```
<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
```

✓ 0s completed at 11:40 PM

```
 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
```

```
X = df # features
y = df.target # target
```

```
# train, test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, train_size= 0.8,
```

```
y # target
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

```
X # df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	t
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	

299	45	1	3	110	264	0	1	132	0	1.2	1	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2
301	57	1	0	130	131	0	1	115	1	1.2	1	1
302	57	0	1	130	236	0	0	174	0	0.0	1	1

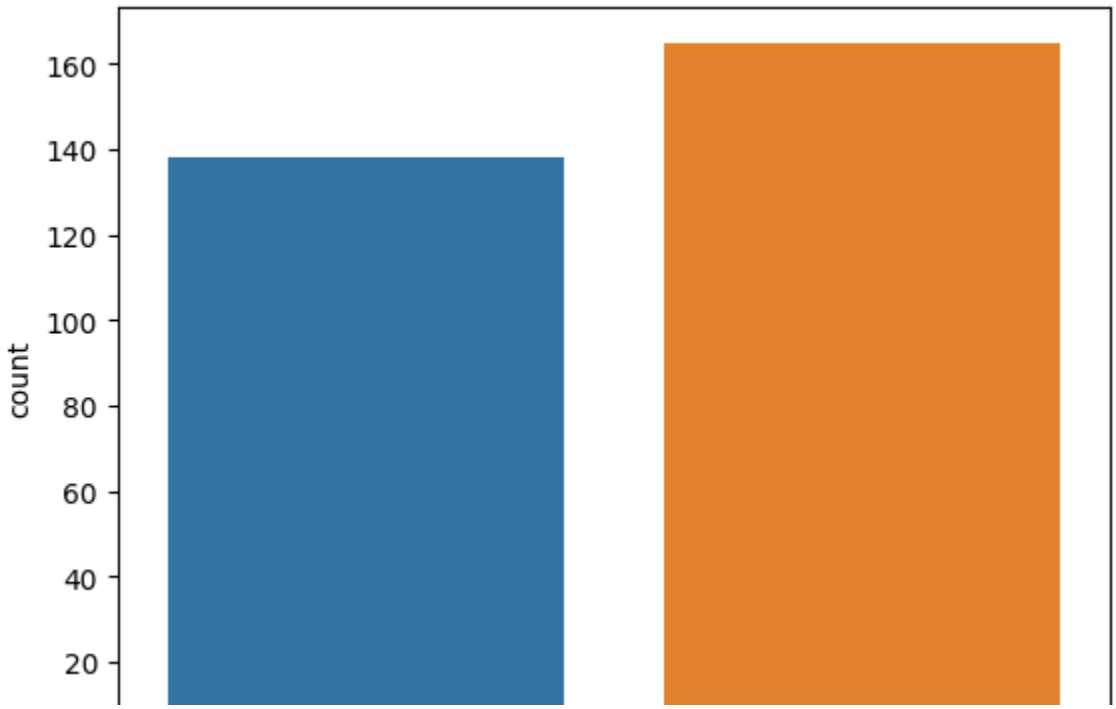
303 rows × 14 columns

```
from sklearn.preprocessing import StandardScaler # standardize dataset
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

```
array([[ 0.9521966 ,  0.68100522,  1.97312292, ..., -0.71442887,
        -2.14887271,  0.91452919],
       [-1.91531289,  0.68100522,  1.00257707, ..., -0.71442887,
        -0.51292188,  0.91452919],
       [-1.47415758, -1.46841752,  0.03203122, ..., -0.71442887,
        -0.51292188,  0.91452919],
       ...,
       [ 1.50364073,  0.68100522, -0.93851463, ...,  1.24459328,
         1.12302895, -1.09345881],
       [ 0.29046364,  0.68100522, -0.93851463, ...,  0.26508221,
         1.12302895, -1.09345881],
       [ 0.29046364, -1.46841752,  0.03203122, ...,  0.26508221,
        -0.51292188, -1.09345881]])
```

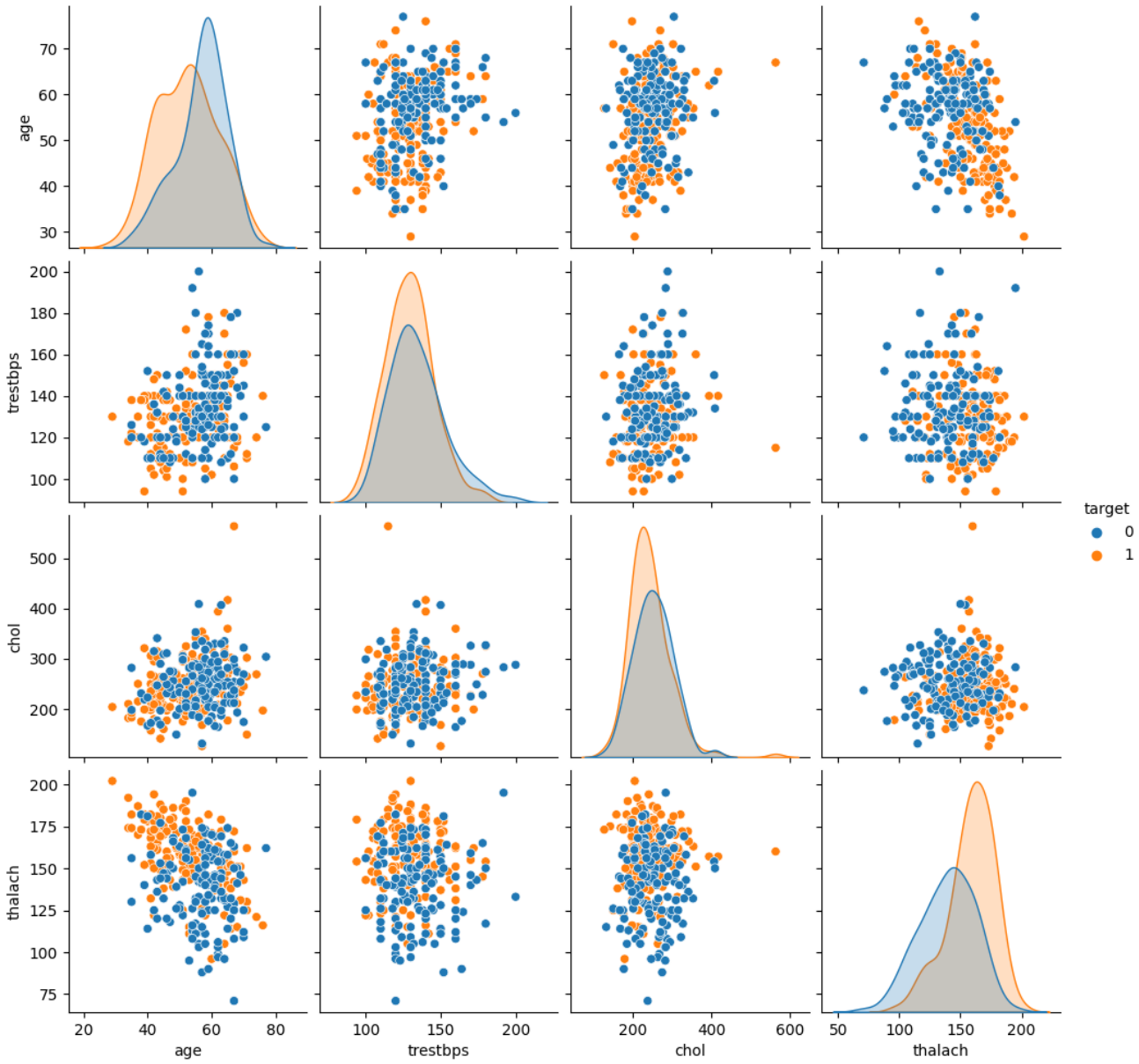
```
sns.countplot(x="target", data = df) # display graph
```

```
<Axes: xlabel='target', ylabel='count'>
```




```
data = ['age', 'trestbps', 'chol', 'thalach', 'target'] # display graph
sns.pairplot(df[data], hue = 'target')
```

```
<seaborn.axisgrid.PairGrid at 0x7fa83627ec70>
```



Description

The dataset heart.csv contain information in relation to heart issues. With all the patient features and health details. The target here is denoted as either 0 or 1. Here 0 indicates the patience have no problems regarding his/her heart. 1 is when a health problem due to heart is apparent. The graph shows the distrubution of the target class in the dataset. With it can be seen the impact of the target value on the dataset.

```
X = df.drop('target', axis = 1) # drops target
y = df['target']
```

```
X_train # input to train on
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	t
260	66	0	0	178	228	1	1	165	1	1.0	1	2	
256	58	1	0	128	259	0	0	130	1	3.0	1	2	
112	64	0	2	140	313	0	1	133	0	0.2	2	0	
110	64	0	0	180	325	0	1	154	1	0.0	2	0	
149	42	1	2	130	180	0	1	150	0	0.0	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
152	64	1	3	170	227	0	0	155	0	0.6	1	0	
204	62	0	0	160	164	0	0	145	0	6.2	0	3	

53	44	0	2	108	141	0	1	175	0	0.6	1	0
294	44	1	0	120	169	0	1	144	1	2.8	0	0
211	61	1	0	120	260	0	1	140	1	3.6	1	1

242 rows × 14 columns

```
naive_bayes = MultinomialNB()
naive_bayes.fit (X_train, y_train)
```

▾ MultinomialNB
MultinomialNB()

```
MultinomialNB(alpha = 1.0, class_prior = None, fit_prior = True)
```

▾ MultinomialNB
MultinomialNB()

```
pred = naive_bayes.predict(X_test)
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,pred)
```

```
array([[20,  8],
       [ 2, 31]])
```

```
nb = GaussianNB() # naive accuracy
nb.fit(X_train, y_train)
nb_score = nb.score(X_test, y_test)
print("accuracy:", nb_score)
```

```
accuracy: 1.0
```

```
print(classification_report(y_test,pred))
```

```
                  precision    recall  f1-score   support

               0       0.91      0.71      0.80        28
               1       0.79      0.94      0.86        33

        accuracy                           0.84        61
       macro avg       0.85      0.83      0.83        61
    weighted avg       0.85      0.84      0.83        61
```

```
from sklearn.linear_model import LogisticRegression # logistic accuracy
```

```
from sklearn.linear_model import LogisticRegression # Logistic accuracy
```

```
model = LogisticRegression()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
1.0
```

```
l = LogisticRegression(random_state=42)
l.fit(X_train, y_train)
l_score = l.score(X_test, y_test)
print("Logistic Regression accuracy:", l_score)
```

```
Logistic Regression accuracy: 1.0
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.neural_network import MLPRegressor
regr = MLPRegressor(hidden_layer_sizes=(6,3), max_iter= 500,
                    random_state= 1234)
regr.fit(X_train, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.
  warnings.warn(
```

▾ MLPRegressor

```
MLPRegressor(hidden_layer_sizes=(6, 3), max_iter=500, random_state=1234)
```

```
y_pred =  regr.predict(X_test)
from sklearn.metrics import mean_squared_error, r2_score
```

```
print('mse=', mean_squared_error(y_test, y_pred))
```

```
mse= 0.590602919102967
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 4


# Define the model
model = keras.Sequential([
    keras.layers.Dense(100, activation='sigmoid', input_shape=(13,)),
    keras.layers.Dense(10, activation='tanh')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=5)

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
```

```
Epoch 1/5
8/8 [==============================] - 0s 2ms/step - loss: 11.9122 - accuracy: 0.0000
Epoch 2/5
8/8 [==============================] - 0s 2ms/step - loss: 7.5257 - accuracy: 0.3843
Epoch 3/5
8/8 [==============================] - 0s 2ms/step - loss: 6.9973 - accuracy: 0.5496
Epoch 4/5
8/8 [==============================] - 0s 2ms/step - loss: 6.9832 - accuracy: 0.5496
Epoch 5/5
8/8 [==============================] - 0s 2ms/step - loss: 6.8516 - accuracy: 0.5496
2/2 [==============================] - 0s 6ms/step - loss: 6.6699 - accuracy: 0.5246
Test accuracy: 0.5245901346206665
```

```
from sklearn.neural_network import MLPClassifier # neural accuracy
n = MLPClassifier(hidden_layer_sizes=(20, 10), max_iter=1000, random_state=42)
n.fit(X_train, y_train)

n_score = n.score(X_test, y_test)
print("accuracy:", n_score)
```

```
accuracy: 0.8688524590163934
```

Analysis

The dataset picked is called heart.csv where as mentioned before contains the data or information of people. The information consists of different features and values in relation to heart disease. The target class here would be if the patient have heart disease or not. After playing with the data the three approaches were Naive Bayes, Logistic Regression and Neural Networks. The three were closes although Logistic Regression has the most accuracy. Neural Networks can more utilised in a more complex tasks, it is that powerful. Neural network a network would have layers and neurons in it. It also has activation function, a function of math to get data from a neuron. With it the values can be more efficient. Though it would do well on a larger size data set. Naive bayes is better on small data and good for multi-class classification though Logistic Regression is good for binary classificaiton. Here the target class is binary. The target here is denoted as either 0 or 1. Therefore, Logistic Regression does a little better.

Colab paid products - Cancel contracts here