Neural Networks & Deep Learning: ICP5

V v s murthy kolla

700729142

Github link: https://github.com/murthykolla/ICP-5.git

Video link
:https://drive.google.com/file/d/1Fa6OPJ8PWGRMjEOF_iMtTOtTvFbGbFbq/view?usp=sharing

```
# 1. Implement Naïve Bayes method using scikit-learn library
# Use dataset available with name glass
# Use train_test_split to create training and testing part
# Evaluate the model on test part using score and
# classification_report(y_true, y_pred)
```

```
#importing requried libraries to
import pandas as pd
#reading the glass.csv
df = pd.read_csv("/Users/vvsmurthykolla/Downloads/NNDL_Code and Data 2/glass.csv")
#print the data in the csv file
print(df)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

```
           RI     Na     Mg     Al     Si     K     Ca     Ba   Fe  Type
0     1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0     1
1     1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     1
2     1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0     1
3     1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0     1
4     1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0     1
..        ...    ...   ...   ...    ...   ...   ...   ...  ...   ...
209   1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0     7
210   1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0     7
211   1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0     7
212   1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0     7
213   1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0     7

[214 rows x 10 columns]
```

```
#print the top values in the dataset
print(df.head())
#print the shape of the dataframe i.e the number of rows and columns
df.shape
#gives the information about the dataframe
df.info()
#prints the description about the dataframe
```

```
        RI     Na     Mg    Al     Si     K    Ca   Ba   Fe  Type
0  1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.0  0.0     1
1  1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.0  0.0     1
2  1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.0  0.0     1
3  1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.0  0.0     1
4  1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.0  0.0     1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

```
#gives the information about the dataframe
df.info()
#prints the description about the dataframe
print(df.describe)
#returns the number of missing values in the dataset
df.isnull().sum()
```

```
<bound method NDFrame.describe of            RI     Na     Mg    Al     Si     K    Ca    Ba   Fe  Type
0     1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0     1
1     1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     1
2     1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0     1
3     1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0     1
4     1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0     1
..        ...    ...   ...   ...    ...   ...   ...   ...  ...   ...
209   1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0     7
210   1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0     7
211   1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0     7
212   1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0     7
213   1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0     7
```

```
# Create a Naive Bayes object
naiveBayesObject = GaussianNB()
# variables x and y are created
x = df.drop(columns=['Type'])
y = df['Type']
#to create train and test data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=4)
#training part of the model
naiveBayesObject.fit(x_train, y_train)
#Predict testing set
y_pred = naiveBayesObject.predict(x_test)
 #performing the accuracy check for the model
print(accuracy_score(y_test, y_pred))
```

```
[214 rows x 10 columns]>
0.4883720930232558
```

```
# 2. Implement linear SVM method using scikit library
# Use the same dataset above
# Use train_test_split to create training and testing part
# Evaluate the model on test part using score and
# classification_report(y_true, y_pred)
```

```
#importing the certain libraries requried for the SVM Model
import pandas as pd
from sklearn.svm._libsvm import predict
#reading the csv file for dataframe
dataFrame = pd.read_csv("/Users/vvsmurthykolla/Downloads/NNDL_Code and Data 2/glass.csv")
#printing the dataframe
print(dataFrame)
```

```
           RI     Na    Mg    Al     Si     K    Ca    Ba   Fe  Type
0     1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0     1
1     1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     1
2     1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0     1
3     1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0     1
4     1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0     1
..        ...    ...   ...   ...    ...   ...   ...   ...  ...   ...
209   1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0     7
210   1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0     7
211   1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0     7
212   1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0     7
213   1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0     7
```

```
#importing the various libraries used by the Linear_SVM method
from sklearn import datasets, metrics
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.model_selection import train_test_split
#Load and return the iris dataset (classification).
dataFrame = datasets.load_iris()
```

```
# to provide labels for x and y axis x - features and y will be labels
X = dataFrame.data
y = dataFrame.target
# splitting  X and Y into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
# training a linear SVM classifier
from sklearn.svm import SVC
svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X_train, y_train)
svm_predictions = svm_model_linear.predict(X_test)
# model accuracy for X_test
accuracy = svm_model_linear.score(X_test, y_test)
#print the the accuracy
print(accuracy)
```

```
[214 rows x 10 columns]
0.9736842105263158
```

In [ ]:
```
# Which algorithm you got better accuracy? Can you justify why?
```

justfication

Linear_SVM method accuracy is o.97 which is greaterthan the Naïve Bayes method accuracy 0.48 .This is beacuse the Naïve Bayes method feature are treated independently and the feature in the lin.ear SVM model are using the non-linear kernel.so linear SVM model have more accuracy is than the Naïve Bayes method