Neural Networks & Deep Learning: ICP1

Name: Prathin Reddy Junnuthula

ID:700741496

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)

```python
# 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)


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score

# load the glass dataset
glass = pd.read_csv("/content/glass.csv")

# split the data into training and testing sets
X = glass.drop("Type", axis=1)
y = glass["Type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# train the Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# make predictions on the test set
y_pred = gnb.predict(X_test)

# evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.5581395348837209

Classification Report:
              precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.43      0.21      0.29        14
           3       0.40      0.67      0.50         3
           5       0.50      0.25      0.33         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.56        43
   macro avg       0.60      0.63      0.59        43
weighted avg       0.55      0.56      0.53        43
```

Neural Networks & Deep Learning: ICP1

Name: Prathin Reddy Junnuthula
ID:700741496

The code implements the Naïve Bayes method using scikit-learn library with the glass dataset. Here's a summary of the steps:

The glass dataset is loaded using pandas.

The data is split into training and testing sets using train_test_split. The test set size is set to 20% of the data with a random state of 42.

A Gaussian Naïve Bayes classifier is trained on the training set.

Predictions are made on the test set using the trained classifier.

The model's performance is evaluated by calculating the accuracy score and printing a classification report.

The accuracy score represents the overall accuracy of the model on the test set, while the classification report provides detailed metrics such as precision, recall, and F1-score for each class in the dataset. These metrics help assess the model's performance in classifying the glass types.

2. Implement linear SVM method using scikit-learn
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)
Which algorithm you got better accuracy? Can you justify why?

Neural Networks & Deep Learning: ICP1

Name: Prathin Reddy Junnuthula
ID:700741496

```
# 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)
# Which algorithm you got better accuracy? Can you justify why?

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

# Load the dataset
df = pd.read_csv("/content/glass.csv")

# Split the dataset into training and testing parts
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train the linear SVM model
svc = SVC(kernel='linear', random_state=0)
svc.fit(X_train, y_train)

# Predict the labels on the test set
y_pred = svc.predict(X_test)

# Evaluate the model
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Classification Report: \n", classification_report(y_test, y_pred))
```

```
Accuracy:  0.5116279069767442
Classification Report:
               precision    recall  f1-score   support

           1       0.36      0.89      0.52         9
           2       0.58      0.37      0.45        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       0.00      0.00      0.00         2
           7       0.86      1.00      0.92         6

    accuracy                           0.51        43
   macro avg       0.38      0.46      0.40        43
weighted avg       0.48      0.51      0.46        43
```

The code implements the linear Support Vector Machine (SVM) method using the scikit-learn library with the same glass dataset. Here's a summary of the steps:

The glass dataset is loaded using pandas.

The data is split into training and testing sets using train_test_split. The test set size is set to 20% of the data with a random state of 0.

A linear SVM model is trained on the training set using the SVC class from scikit-learn. The 'linear' kernel is used for a linear SVM.

Neural Networks & Deep Learning: ICP1
Name: Prathin Reddy Junnuthula
ID:700741496

Predictions are made on the test set using the trained SVM model.

The model's performance is evaluated by calculating the accuracy score and printing a classification report.

The accuracy score represents the overall accuracy of the SVM model on the test set, while the classification report provides detailed metrics such as precision, recall, and F1-score for each class in the dataset.

To compare the accuracy of the Naïve Bayes method and linear SVM method, you can compare the accuracy scores obtained from both models. The model with a higher accuracy score would be considered better in terms of accuracy for this dataset. However, the choice of the better algorithm may depend on various factors such as the nature of the data, the distribution of classes, and the problem at hand. It is recommended to evaluate the performance of different algorithms on multiple metrics and consider the specific requirements of the problem before deciding.

3. Implement Linear Regression using scikit-learn
a) Import the given “Salary_Data.csv”
b) Split the data in train test partitions, such that 1/3 of the data is reserved as test subset.
c) Train and predict the model.
d) Calculate the mean squared error.
e) Visualize both train and test data using scatter plot.

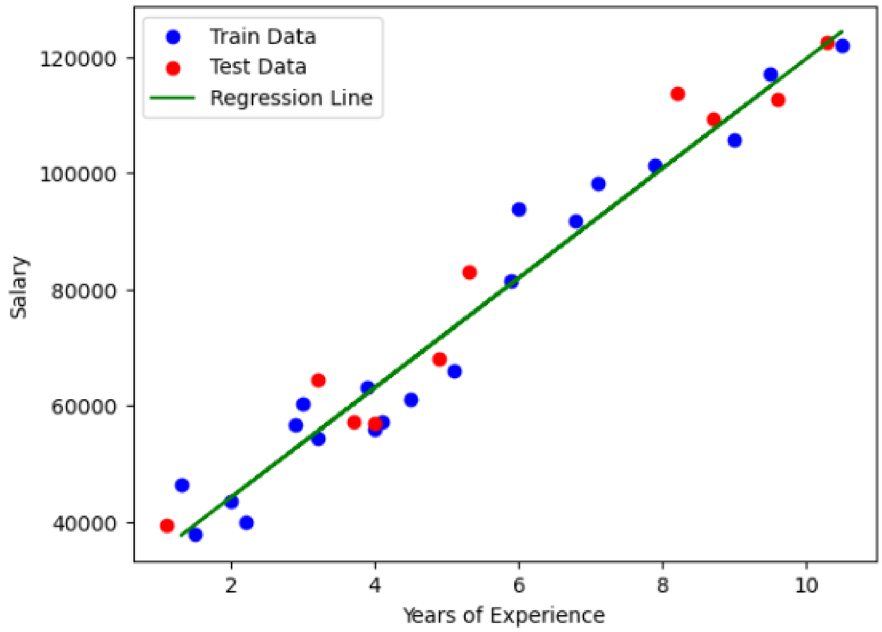
Neural Networks & Deep Learning: ICP1

Name: Prathin Reddy Junnuthula
ID:700741496

```python
# 3.Implement Linear Regression using scikit-learn
# Import the given "Salary_Data.csv"
# Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
# Train and predict the model.
# Calculate the mean_squared error.
# Visualize both train and test data using scatter plot


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Import the given "Salary_Data.csv"
data = pd.read_csv(r"C:\Users\priya\Downloads\Salary_Data.csv")

# Split the data into input features (X) and target variable (y)
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

#  Split the data into train_test partitions
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=42)

#  Train and predict the model
model = LinearRegression()
model.fit(X_train, y_train)
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

#  Calculate the mean squared error
mse_train = mean_squared_error(y_train, y_train_pred)
mse_test = mean_squared_error(y_test, y_test_pred)
print("Mean Squared Error (Train):", mse_train)
print("Mean Squared Error (Test):", mse_test)

#  Visualize both train and test data using scatter plot
plt.scatter(X_train, y_train, color='blue', label='Train Data')
plt.scatter(X_test, y_test, color='red', label='Test Data')
plt.plot(X_train, y_train_pred, color='green', label='Regression Line')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.legend()
plt.show()
```

Neural Networks & Deep Learning: ICP1
Name: Prathin Reddy Junnuthula
ID:700741496

```
Mean Squared Error (Train): 29793161.082422983
Mean Squared Error (Test): 35301898.887134895
```



a) We import the necessary libraries: pandas for data manipulation, numpy for array operations, matplotlib.pyplot for visualization, and the required scikit-learn modules.

b) We read the "Salary_Data.csv" file into a panda DataFrame.

c) We split the data into input features X and the target variable y. Then, we split the data into training and test sets using train_test_split(), with a test size of 1/3 and a random state of 42.

d) We create an instance of the Linear Regression model, train it on the training data using fit(), and make predictions on both the training and test data using predict().

e) We calculate the mean squared error (MSE) using the mean_squared_error() function and print the values.

Finally, we visualize the training and test data points as scatter plots and plot the regression line using plt.scatter() and plt.plot(). The plot is displayed using plt.show().

GitHub URL:

https://github.com/Prathin-offi/NNDL_ICP-1.git

Neural Networks & Deep Learning: ICP1

Name: Prathin Reddy Junnuthula
ID:700741496

Video URL:

https://drive.google.com/file/d/1uw91OMojumlXDaXnqj1XZAT0a47OME04/view?usp=sharing