1. Write a program to implement Simple Linear Regression Using Python ¶

In [74]:
```
import pandas as pd
import numpy as np
```

In [75]:
```
dataset = pd.read_csv('student_scores.csv')
```

In [76]:
```
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,1].values
```

In [77]:
```
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=0)
```

In [78]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
```

Out[78]:
```
▾ LinearRegression
LinearRegression()
```

In [79]:
```
y_pred = regressor.predict(x_test)
```

In [80]:
```
from sklearn import metrics
print("mean absolute error : ",metrics.mean_absolute_error(y_test,y_pred))
print("mean squared error : ",metrics.mean_squared_error(y_test,y_pred))
print("Root mean squared error : ",np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
mean absolute error :  4.183859899002975
mean squared error :  21.5987693072174
Root mean squared error :  4.6474476121003665
```

In [81]:
```
regressor.score(x_test,y_test)
```

Out[81]:
```
0.9454906892105356
```

2. Write a program to implement KNN Using Python

In [82]:
```
import pandas as pd
import numpy as np
```

In [83]:
```
dataset = pd.read_csv('User_Data.csv')
```

In [84]:
```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [85]:
```
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=0)
```

In [86]:
```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [87]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(xtrain, y_train)
```

Out[87]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [88]:
```
y_pred = classifier.predict(xtest)
```

In [89]:

```
from sklearn.metrics import classification_report,confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[89]:

```
array([[55,  3],
       [ 1, 21]], dtype=int64)
```

In [90]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80
```

3. Write a program to implement Support Vector Machine (SVM) Using Python

In [91]:

```
import pandas as pd
import numpy as np
```

In [92]:

```
dataset = pd.read_csv('User_Data.csv')
```

In [93]:

```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [94]:

```
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=0)
```

In [95]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [96]:

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear',random_state = 0)
classifier.fit(xtrain, y_train)
```

Out[96]:

```
▾ SVC
SVC(kernel='linear', random_state=0)
```

In [97]:

```
y_pred = classifier.predict(xtest)
```

In [98]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[98]:

```
array([[57,  1],
       [ 6, 16]], dtype=int64)
```

In [99]:

```
from sklearn.metrics import accuracy_score
print("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.9125
```

4. Write a program to implement Multiple Linear Regression Using Python

In [100]:

```
import pandas as pd
import numpy as np
```

In [101]:

```
dataset = pd.read_csv('house_data.csv')
```

In [102]:

```python
x = dataset.iloc[:,[5]].values
y = dataset.iloc[:,-1].values
```

In [103]:

```python
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=0)
```

In [104]:

```python
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[104]:

```
▾ LinearRegression
LinearRegression()
```

In [105]:

```python
y_pred = regressor.predict(x_test)
```

In [106]:

```python
from sklearn import metrics
print('Mean Absolute Error : ',metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error : ',metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error : ',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error :  188654.74349853912
Mean Squared Error :  76981618783.81517
Root Mean Squared Error :  277455.6158808381
```

In [107]:

```python
regressor.score(x_test,y_test)
```

Out[107]:

```
0.35355693552757517
```

5. Write a program to demonstrate Classification Report

In [108]:

```python
import pandas as pd
import numpy as np
```

In [109]:

```python
dataset = pd.read_csv('User_Data.csv')
```

In [110]:

```python
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [111]:

```python
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=0)
```

In [112]:

```python
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [113]:

```python
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(xtrain,y_train)
```

Out[113]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [114]:

```python
y_pred = classifier.predict(x_test)
```

In [115]:

```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[115]:

```
array([[ 0, 58],
       [ 0, 22]], dtype=int64)
```

In [207]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80
```

6. Write program for linear regression and find parameters like Sum of Squared Errors (SSE), Total Sum of Squares (SST), R2, Adjusted R2etc.

In [117]:
```
import pandas as pd
import numpy as np
```

In [118]:
```
dataset = pd.read_csv('student_scores.csv')
```

In [119]:
```
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,1].values
```

In [120]:
```
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=0)
```

In [121]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Out[121]:
```
▾ LinearRegression
LinearRegression()
```

In [122]:
```
y_pred = regressor.predict(x_test)
```

In [123]:
```
# calculate sse
sse = np.sum((y_test - y_pred)**2)
print(sse)
```

```
107.99384653608699
```

In [124]:
```
# calculate ssr
ssr = np.sum((y_pred - y.mean()**2))
print(ssr)
```

```
-13037.692741939658
```

In [125]:
```
# calculate sst
sst = ssr + sse
print(sst)
```

```
-12929.698895403571
```

7. Write a program to demonstrate Confusion Matrix

In [126]:
```
import pandas as pd
import numpy as np
```

In [127]:
```
dataset = pd.read_csv('User_Data.csv')
```

In [128]:
```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [129]:
```
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=0)
```

In [130]:

```python
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [131]:

```python
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy',random_state=0)
classifier.fit(xtrain,y_train)
```

Out[131]:

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [132]:

```python
y_pred = classifier.predict(xtest)
```

In [133]:

```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[133]:

```
array([[53,  5],
       [ 3, 19]], dtype=int64)
```

In [134]:

```python
from sklearn.metrics import accuracy_score
print("accuracy_score : ",accuracy_score(y_test, y_pred))
```

```
accuracy_score :  0.9
```

8. Write a program to demonstrate Precision , Recall, F1-Score

In [135]:

```python
import pandas as pd
import numpy as np
```

In [136]:

```python
dataset = pd.read_csv('User_Data.csv')
```

In [137]:

```python
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [138]:

```python
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=0)
```

In [139]:

```python
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [140]:

```python
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy',random_state=0)
classifier.fit(xtrain,y_train)
```

Out[140]:

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [141]:

```python
y_pred = classifier.predict(xtest)
```

In [142]:

```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[142]:

```
array([[53,  5],
       [ 3, 19]], dtype=int64)
```

In [143]:

```python
# PRINT PRECISION, RECALL(SENSITIVITY), F1-SCORE
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
```

In [144]:

```python
# calculate accuracy
accuracy = accuracy_score(y_test,y_pred)
print("Accuracy : ",accuracy)
```

```
Accuracy :  0.9
```

In [145]:
```
# calculate precision
precision = precision_score(y_test,y_pred)
print("Precision : ",precision)
```

```
Precision :  0.7916666666666666
```

In [146]:
```
# calculate f1-score
f1 = f1_score(y_test,y_pred)
print("f1-Score : ",f1)
```

```
f1-Score :  0.8260869565217391
```

9. Write a program to implement Decision Trees Using Python

In [147]:
```
import pandas as pd
import numpy as np
```

In [148]:
```
dataset = pd.read_csv('User_Data.csv')
```

In [149]:
```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [150]:
```
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=0)
```

In [151]:
```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [152]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy',random_state=0)
classifier.fit(xtrain, y_train)
```

Out[152]:
```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [153]:
```
y_pred = classifier.predict(xtest)
```

In [154]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[154]:
```
array([[53,  5],
       [ 3, 19]], dtype=int64)
```

In [155]:
```
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.9
```

10. Write a program to implement Logistic Regression Using Python

In [156]:
```
import pandas as pd
import numpy as np
```

In [157]:
```
dataset = pd.read_csv("User_Data.csv")
```

In [158]:
```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [159]:
```
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=0)
```

In [160]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [161]:

```
from sklearn.linear_model import LogisticRegression
regressor = LogisticRegression(random_state=0)
regressor.fit(xtrain, y_train)
```

Out[161]:

```
▾       LogisticRegression
LogisticRegression(random_state=0)
```

In [162]:

```
y_pred = regressor.predict(xtest)
```

In [163]:

```
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test,y_pred))
```

```
Accuracy :  0.925
```

11. Write a program to implement decision tree algorithm to classify the iris Dataset. Print both correct and wrong predictions.

In [164]:

```
from sklearn.datasets import load_iris
```

In [165]:

```
dataset = load_iris()
x = dataset.data
y = dataset.target
```

In [166]:

```
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=0)
```

In [167]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [168]:

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state=0,criterion='gini')
classifier.fit(xtrain,y_train)
```

Out[168]:

```
▾       DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [169]:

```
y_pred = classifier.predict(xtest)
```

In [170]:

```
from sklearn.metrics import accuracy_score
print("Test data Accuracy : ",accuracy_score(y_test,y_pred))
```

```
Test data Accuracy :  1.0
```

In [171]:

```
i=0
print('Original Label ',' Predicted Label ',' Correct/Wrong')
for label in y_test:
    print(label ,"\t\t", y_pred[i], end="\t\t")
    if(label == y_pred[i]):
        print('     Correct')
    else:
        print('     Wrong')
    i = i+1
```

```
Original Label   Predicted Label   Correct/Wrong
2                2                    Correct
1                1                    Correct
0                0                    Correct
2                2                    Correct
0                0                    Correct
2                2                    Correct
0                0                    Correct
1                1                    Correct
1                1                    Correct
1                1                    Correct
2                2                    Correct
1                1                    Correct
1                1                    Correct
1                1                    Correct
1                1                    Correct
0                0                    Correct
1                1                    Correct
1                1                    Correct
0                0                    Correct
0                0                    Correct
2                2                    Correct
1                1                    Correct
0                0                    Correct
0                0                    Correct
2                2                    Correct
0                0                    Correct
0                0                    Correct
1                1                    Correct
1                1                    Correct
0                0                    Correct
```

12. Write a program to implement the naïve Bayesian classifier for a sample training dataset stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets

In [172]:

```
import pandas as pd
import numpy as np
```

In [173]:

```
dataset = pd.read_csv('User_Data.csv')
```

In [174]:

```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [175]:

```
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=0)
```

In [176]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [177]:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(xtrain, y_train)
```

Out[177]:

```
▾ GaussianNB
GaussianNB()
```

In [178]:

```
y_pred = classifier.predict(xtest)
```

In [179]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[179]:

```
array([[55,  3],
       [ 4, 18]], dtype=int64)
```

In [180]:

```
from sklearn.metrics import accuracy_score
print("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.9125
```

13. Write a program to implement Naive Bayes Classifier Using Python

In [181]:

```
import pandas as pd
import numpy as np
```

In [182]:

```
dataset = pd.read_csv('User_Data.csv')
```

In [183]:

```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [184]:

```
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=0)
```

In [185]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [186]:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(xtrain, y_train)
```

Out[186]:

```
▾ GaussianNB
GaussianNB()
```

In [187]:

```
y_pred = classifier.predict(xtest)
```

In [188]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[188]:

```
array([[55,  3],
       [ 4, 18]], dtype=int64)
```

In [189]:

```
from sklearn.metrics import accuracy_score
print("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.9125
```

14. Write a program to implement k-Nearest Neighbour algorithm to classify the iris dataset. Print both correct and wrong predictions.

In [190]:

```
import pandas as pd
import numpy as np
```

In [191]:

```
dataset = pd.read_csv("IRIS.csv")
```

In [192]:

```
x = dataset.iloc[:,:-1]
y = dataset.iloc[:,-1]
```

In [193]:

```
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=0)
```

In [194]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
```

Out[194]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [195]:
```
y_pred = classifier.predict(x_test)
```

In [196]:
```
i=0
print('Original Label  ',' Predicted Label ',' Correct/Wrong  ')
for label in y_test:
    print(label ,y_pred[i],end="")
    if(label == y_pred[i]):
        print('            Correct')
    else:
        print('            Wrong')
    i = i+1
```

```
Original Label    Predicted Label   Correct/Wrong
Iris-virginica Iris-virginica            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-setosa Iris-setosa            Correct
Iris-virginica Iris-virginica            Correct
Iris-setosa Iris-setosa            Correct
Iris-virginica Iris-virginica            Correct
Iris-setosa Iris-setosa            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-virginica Iris-virginica            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-versicolor Iris-virginica            Wrong
Iris-setosa Iris-setosa            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-setosa Iris-setosa            Correct
Iris-setosa Iris-setosa            Correct
Iris-virginica Iris-virginica            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-setosa Iris-setosa            Correct
Iris-setosa Iris-setosa            Correct
Iris-virginica Iris-virginica            Correct
Iris-setosa Iris-setosa            Correct
Iris-setosa Iris-setosa            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-versicolor Iris-versicolor            Correct
Iris-setosa Iris-setosa            Correct
```

15. Implement simple KNN using Euclidean distance in python

In [197]:
```
import pandas as pd
import numpy as np
```

In [198]:
```
dataset = pd.read_csv('User_Data.csv')
```

In [199]:
```
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
```

In [200]:
```
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=0)
```

In [201]:
```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(x_train)
xtest = sc_x.transform(x_test)
```

In [202]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(xtrain, y_train)
```

Out[202]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [203]:

```
y_pred = classifier.predict(xtest)
```

In [204]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[204]:

```
array([[55,  3],
       [ 1, 21]], dtype=int64)
```

In [205]:

```
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test, y_pred))
```

```
Accuracy :  0.95
```

In [206]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        58
           1       0.88      0.95      0.91        22

    accuracy                           0.95        80
   macro avg       0.93      0.95      0.94        80
weighted avg       0.95      0.95      0.95        80
```