Linear Regression Assignment

Problem Statement: You work in XYZ Company as a Python Data Scientist. The company officials have collected some data on salaries based on year of experience and wish for you to create a model from it. Dataset: data.csv Tasks to be performed:

1. Load the dataset using pandas
2. Extract data fromYearsExperience column is a variable named X
3. Extract data from salary column is a variable named Y
4. Divide the dataset into two parts for training and testing in 66% and 33% proportion
5. Create and train LinearRegression Model on training set
6. Make predictions based on the testing set using the trained model
7. Check the performance by calculating the r2 score of the model

In [28]:
```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assighnment pdf s')
```

In [29]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
data=pd.read_csv('data.csv')
data.head()
```

Out[29]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [30]:
```
X=data[['YearsExperience']]
Y=data[['Salary']]
```

In [31]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

In [32]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.33)
```

In [33]:
```
lr.fit(X_train,Y_train)
```

Out[33]:
```
LinearRegression()
```

In [34]:
```
lr.predict(X_test)
```

Out[34]:
```
array([[ 44568.93391277],
       [ 53109.2606687 ],
       [108146.9219847 ],
       [ 55956.03625401],
       [ 62598.51261974],
       [ 75883.46535119],
       [ 81577.01652181],
       [115738.32354553],
       [125227.57549657],
       [ 73985.61496098]])
```

In [35]:
```
lr.score(X_test, Y_test)
```

Out[35]:
```
0.9791612020545786
```

Logistic Regression Assignment

Problem Statement: You work in XYZ Company as a Python Data Scientist. The company officials have collected some data on Health parameter based on Diabetes and wish for you to create a model from it. Dataset: diabetes.csv Tasks to be performed:

1. Load the dataset using pandas
2. Extract data fromOutcome column is a variable named Y
3. Extract data from every column except Outcome column in a variable named X
4. Divide the dataset into two parts for training and testing in 70% and 30% proportion
5. Create and train Logistic Regression Model on training set
6. Make predictions based on the testing set using the trained model
7. Check the performance by calculating the confusion matrix and accuracy score of the model

In [36]:
```
df=pd.read_csv('diabetes.csv')
df.head()
```

Out[36]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [37]:
```
X=df[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Dia
Y=df[['Outcome']]
```

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

In [38]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3)
```

In [39]:
```
from sklearn.linear_model import LogisticRegression
```

In [40]:
```
lr=LogisticRegression()
```

In [41]:
```
lr.fit(X_train,Y_train)
```

```
C:\Users\veena\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataCo
nversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\veena\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814:
ConvergenceWarning: lbfgs failed to converge (status=1):
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(
```

Out[41]:
```
LogisticRegression()
```

In [42]:
```
lr.predict(X_test)
```

Out[42]:
```
array([1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0], dtype=int64)
```

In [34]:
```
lr.score(X_test,Y_test)
```

Out[34]:
```
0.8008658008658008
```

Decision Tree Assignment

Problem Statement: You work in XYZ Company as a Python Data Scientist. The company officials have collected some data on salaries based on year of experience and wish for you to create a model from it. Dataset: diabetes.csv Tasks to be performed:

1. Load the dataset using pandas
2. Extract data fromOutcome column is a variable named Y
3. Extract data from every column except Outcome column in a variable named X
4. Divide the dataset into two parts for training and testing in 70% and 30% proportion
5. Create and train Decision Tree Model on training set
6. Make predictions based on the testing set using the trained model
7. Check the performance by calculating the confusion matrix and accuracy score of the model

In [43]:
```
df1=pd.read_csv('diabetes.csv')
df1.head()
```

Out[43]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [44]:
```
X=df1[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Di
Y=df1[['Outcome']]
```

In [45]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3)
```

In [46]:
```
from sklearn import tree
```

In [47]:
```
tr=tree.DecisionTreeClassifier()
```

In [48]:
```
tr.fit(X_test, Y_test)
```

Out[48]:
```
DecisionTreeClassifier()
```

In [49]:
```
tr.predict(X_test)
```

Out[49]:
```
array([1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0], dtype=int64)
```

In [50]:
```
tr.score(X_test, Y_test)
```

Out[50]:
```
1.0
```

In [51]:
```
y_pred=tr.predict(X_test)
```

In [52]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,y_pred)
cm
```

Out[52]:
```
array([[140,   0],
       [  0,  91]], dtype=int64)
```

In [ ]:

Random Forest Assignment

Problem Statement: You work in XYZ Company as a Python Data Scientist. The company officials have collected some data on salaries based on year of experience and wish for you to create a model from it. Dataset: diabetes.csv Tasks to be performed:

1. Load the dataset using pandas
2. Extract data fromOutcome column is a variable named Y
3. Extract data from every column except Outcome column in a variable named X.
4. Divide the dataset into two parts for training and testing in 70% and 30% proportion
5. Create and train Random Forest Model on training set
6. Make predictions based on the testing set using the trained model
7. Check the performance by calculating the confusion matrix and accuracy score of the model

In [1]:
```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assighnment pdf s')
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
df2=pd.read_csv('diabetes.csv')
df2.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [5]:
```
df2.isnull().sum()
```

Out[5]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [4]:
```
a=df2[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Di
b=df2[['Outcome']]
```

In [7]:
```
from sklearn.model_selection import train_test_split
```

In [8]:
```
a_train, b_train, a_test, b_test = train_test_split(a,b,test_size=0.3)
```

In [9]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

```
rfc
```

```
Out[9]:  ▾ RandomForestClassifier
         RandomForestClassifier()
```

```
In [13]: df2.shape

Out[13]: (768, 9)
```

```
In [11]: rfc.fit(a_train,b_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [11], in <cell line: 1>()
----> 1 rfc.fit(a_train,b_train)

File ~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:346, in BaseForest.
fit(self, X, y, sample_weight)
    344 if issparse(y):
    345     raise ValueError("sparse multilabel-indicator for y is not supporte
d.")
--> 346 X, y = self._validate_data(
    347     X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
    348 )
    349 if sample_weight is not None:
    350     sample_weight = _check_sample_weight(sample_weight, X)

File ~\anaconda3\lib\site-packages\sklearn\base.py:554, in BaseEstimator._validate
_data(self, X, y, reset, validate_separately, **check_params)
    552         y = check_array(y, input_name="y", **check_y_params)
    553     else:
--> 554         X, y = check_X_y(X, y, **check_params)
    555     out = X, y
    557 if not no_val_X and check_params.get("ensure_2d", True):

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:1122, in check_X_y
(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, e
nsure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numer
ic, estimator)
   1104 X = check_array(
   1105     X,
   1106     accept_sparse=accept_sparse,
   (...)
   1117     input_name="X",
   1118 )
   1120 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=
estimator)
-> 1122 check_consistent_length(X, y)
   1124 return X, y

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:397, in check_consi
stent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples: %r"
    399         % [int(l) for l in lengths]
    400     )

ValueError: Found input variables with inconsistent numbers of samples: [537, 231]
```

In [ ]:

Naïve-Bayes Assignment

Problem Statement: You work in XYZ Company as a Python Data Scientist. The company officials have collected some data on salaries based on year of experience and wish for you to create a model from it. Dataset: diabetes.csv Tasks to be performed:

1. Load the dataset using pandas
2. Extract data fromOutcome column is a variable named Y
3. Extract data from every column except Outcome column in a variable named X
4. Divide the dataset into two parts for training and testing in 70% and 30% proportion
5. Create and train Naïve Bayes Model on training set
6. Make predictions based on the testing set using the trained model
7. Check the performance by calculating the confusion matrix and accuracy score of the model

In [ ]:

In [3]:
```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assighnment pdf s')
```

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df3=pd.read_csv('diabetes.csv')
df3.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [5]:
```
x= df3.drop('Outcome',axis='columns')
y=df3.Outcome
```

In [7]:
```
x.head()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [8]:
```
y.head()
```

Out[8]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [10]:
```
x_train, x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [11]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

In [12]:
```
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [13]:
```
from sklearn.naive_bayes import GaussianNB
gsb=GaussianNB()
gsb.fit(x_train, y_train)
```

Out[13]:
```
GaussianNB()
```

In [21]:
```
y_pred=gsb.predict(x_test)
```

In [24]:
```
gsb.predict(x_test)
```

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

In [23]:
```
gsb.score(x_test,y_test)
```

Out[23]:
```
0.7272727272727273
```

In [26]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[26]:
```
array([[128,  29],
       [ 34,  40]], dtype=int64)
```

In [ ]: