**Sample Part-B Case Study**

**Problem Statement/ Scenario Description:** We will be focusing on the company analysis who are considered as startups. This case study includes a complete analysis of salary regression model, implementing logistic regression and creating a classification model to identify the fraudulent companies

**Question 1**:

Creating a linear regression model of salary and years of experience following the supervised machine learning algorithm. This will involve Conceptual Understanding of supervised learning algorithm linear regression and logistic regression and difference between them.

**Answer**:

Dataset includes two variables namely salary and years of experience. The main motive is to understand the relationship between both. To understand linear regression, we need to understand the relationship between both salary and years of experience.

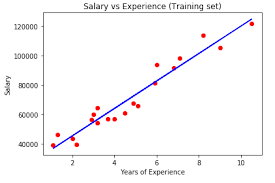
Step 1: Implement the libraries which reads the dataset and split them accordingly as training and testing dataset.

|  |
| --- |
| # Simple Linear Regression  # Importing the libraries  import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  # Importing the datasets    datasets = pd.read\_csv('Salary\_Data.csv')  X = datasets.iloc[:, :-1].values  Y = datasets.iloc[:, 1].values  # Splitting the dataset into the Training set and Test set  from sklearn.model\_selection import train\_test\_split  X\_Train, X\_Test, Y\_Train, Y\_Test = train\_test\_split(X, Y, test\_size = 1/3, random\_state = 0)  # Fitting Simple Linear Regression to the training set  from sklearn.linear\_model import LinearRegression  regressor = LinearRegression()  regressor.fit(X\_Train, Y\_Train)  # Predicting the Test set result ￼  Y\_Pred = regressor.predict(X\_Test) |

Step 2: Visualize the regressor model to understand the best fit line or is there any chances to have a look on the overfit line.

|  |
| --- |
| # Visualising the Training set results  plt.scatter(X\_Train, Y\_Train, color = 'red')  plt.plot(X\_Train, regressor.predict(X\_Train), color = 'blue')  plt.title('Salary vs Experience (Training Set)')  plt.xlabel('Years of experience')  plt.ylabel('Salary')  plt.show()  # Visualising the Test set results  plt.scatter(X\_Test, Y\_Test, color = 'red')  plt.plot(X\_Train, regressor.predict(X\_Train), color = 'blue')  plt.title('Salary vs Experience (Training Set)')  plt.xlabel('Years of experience')  plt.ylabel('Salary')  plt.show() |

**Expected Output:**



**Question 2:**

Applying logistic regression to find out statistics on an advertising dataset for a company. We will be working with an advertising data set of startup company, indicating whether an internet user clicked on an Advertisement. We will try to create a model that will predict whether they will click on an ad based off the features of that user.

**Answer**:

Dataset includes two variables namely salary and years of experience. The main motive is to understand the relationship between both. To understand linear regression, we need to understand the relationship between both salary and years of experience.

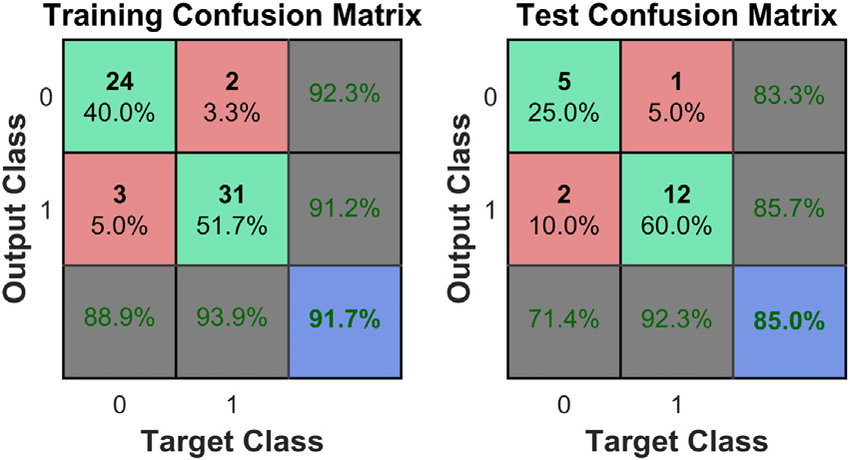
Step 1: Implement the libraries which reads the dataset and split them accordingly as training and testing dataset.

|  |
| --- |
| #Importing Important Libraries  import matplotlib.pyplot as plt  import seaborn as sns  import pandas as pd  import numpy as np  #Reading the data from the csv file.  ad\_data = pd.read\_csv('advertising.csv')  #Checking the head, info and description of the data.  ad\_data.head()  ad\_data.info()  ad\_data.describe()  #Visualizing the data  #Creating a histogram of the age.  sns.distplot(ad\_data['Age'],kde=False,bins=30)  #Creating a jointplot showing Area Income versus Age.  sns.jointplot(y='Area Income',x='Age',data=ad\_data)  #Creating a jointplot showing the kde distributions of Daily Time spent on site vs. Age.  sns.jointplot(y='Daily Time Spent on Site',x='Age',data=ad\_data,kind='kde', color='red')  #Creating a jointplot of 'Daily Time Spent on Site' vs. 'Daily Internet Usage'.  sns.jointplot(x='Daily Time Spent on Site',y='Daily Internet Usage',data=ad\_data,color='green')  #creating a pairplot with the hue defined by the 'Clicked on Ad' column feature.  sns.pairplot(ad\_data,hue='Clicked on Ad')  #Spliting the data into training set and testing set using train\_test\_split.  from sklearn.model\_selection import train\_test\_split  X = ad\_data[['Daily Time Spent on Site', 'Age', 'Area Income',  'Daily Internet Usage', 'Male']]  y = ad\_data['Clicked on Ad']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=46) |

Step 2: Implement the logistic regression model to define the accuracy rate of data.

|  |
| --- |
| #Importing the logistic regression module and fitting the data  from sklearn.linear\_model import LogisticRegression  lg = LogisticRegression()  lg.fit(X\_train,y\_train)  #Predictions and Evaluations  predictions = lg.predict(X\_test)  #Creating a classification report for the model.  from sklearn.metrics import confusion\_matrix  confusion\_matrix(y\_test,predictions)  # |

**Expected Output:**



**Question 3:**

Create the multiple linear regression of the list of companies which are considered as a startup company

**Answer**:

The data includes following attributes

R&D Spend

Administration

Marketing Spend

State-Profit

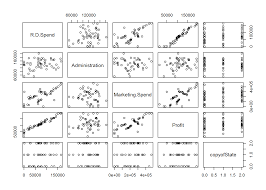
Step 1: Import the modules and create the training and testing dataset to implement the regression analysis.

|  |
| --- |
| # Multiple Linear Regression  import numpy as np  import pandas as pd  # Importing the datasets  datasets = pd.read\_csv('50\_Startups.csv')  X = datasets.iloc[:, :-1].values  Y = datasets.iloc[:, 4].values  # Encoding categorical data  # Encoding the Independent Variable  from sklearn.preprocessing import LabelEncoder, OneHotEncoder  labelencoder\_X = LabelEncoder()  X[:, 3] = labelencoder\_X.fit\_transform(X[:, 3])  onehotencoder = OneHotEncoder(categorical\_features = [3])  X = onehotencoder.fit\_transform(X).toarray()  # Avoiding the Dummy Variable Trap  X = X[:, 1:]  # Splitting the dataset into the Training set and Test set  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)  # Fitting the Multiple Linear Regression in the Training set  from sklearn.linear\_model import LinearRegression  regressor = LinearRegression()  regressor.fit(X\_Train, Y\_Train)  # Predicting the Test set results  Y\_Pred = regressor.predict(X\_Test) |

Step 2: Building the optimal model and fitting them in the regression model to create the complete analysis is mentioned below:

|  |
| --- |
| # Building the optimal model using Backward Elimination  import statsmodels.formula.api as sm  X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)  X\_Optimal = X[:, [0,1,2,3,4,5]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  X\_Optimal = X[:, [0,1,2,4,5]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  X\_Optimal = X[:, [0,1,4,5]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  X\_Optimal = X[:, [0,1,4]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  # Fitting the Multiple Linear Regression in the Optimal Training set  X\_Optimal\_Train, X\_Optimal\_Test = train\_test\_split(X\_Optimal,test\_size = 0.2, random\_state = 0)  regressor.fit(X\_Optimal\_Train, Y\_Train)  # Predicting the Optimal Test set results  Y\_Optimal\_Pred = regressor.predict(X\_Optimal\_Test) |

**Expected Output:**



**Question 4:**

Applying classification model which helps us to show the discrete classification of all the information which is needed for creating a classification model

**Answer**:

Step 1: Import the modules and create the training and testing dataset to implement the regression analysis.

|  |
| --- |
| # Multiple Linear Regression  import numpy as np  import pandas as pd  # Importing the datasets  datasets = pd.read\_csv('50\_Startups.csv')  X = datasets.iloc[:, :-1].values  Y = datasets.iloc[:, 4].values  # Encoding categorical data  # Encoding the Independent Variable  from sklearn.preprocessing import LabelEncoder, OneHotEncoder  labelencoder\_X = LabelEncoder()  X[:, 3] = labelencoder\_X.fit\_transform(X[:, 3])  onehotencoder = OneHotEncoder(categorical\_features = [3])  X = onehotencoder.fit\_transform(X).toarray()  # Avoiding the Dummy Variable Trap  X = X[:, 1:]  # Splitting the dataset into the Training set and Test set  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)  # Fitting the Multiple Linear Regression in the Training set  from sklearn.linear\_model import LinearRegression  regressor = LinearRegression()  regressor.fit(X\_Train, Y\_Train)  # Predicting the Test set results  Y\_Pred = regressor.predict(X\_Test) |

Step 2: Building the optimal model and fitting them in the regression model to create the complete analysis is mentioned below:

|  |
| --- |
| # Building the optimal model using Backward Elimination  import statsmodels.formula.api as sm  X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)  X\_Optimal = X[:, [0,1,2,3,4,5]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  X\_Optimal = X[:, [0,1,2,4,5]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  X\_Optimal = X[:, [0,1,4,5]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  X\_Optimal = X[:, [0,1,4]]  regressor\_OLS = sm.OLS(endog = Y, exog = X\_Optimal).fit()  regressor\_OLS.summary()  # Fitting the Multiple Linear Regression in the Optimal Training set  X\_Optimal\_Train, X\_Optimal\_Test = train\_test\_split(X\_Optimal,test\_size = 0.2, random\_state = 0)  regressor.fit(X\_Optimal\_Train, Y\_Train)  # Predicting the Optimal Test set results  Y\_Optimal\_Pred = regressor.predict(X\_Optimal\_Test) |

**Expected Output:**

