Experiment Number : 3 - Predicting missing data values using regression modeling

---

Batch: FDS-2 **Roll Number: 16010422234** **Name: Chandana Ramesh Galgali**

Aim of the Experiment: Predict missing data values using regression modeling.

Program/ Steps:

Activity 0:

Problem statement:
Let's predict temperature through interpolation techniques . The Data given points are as follows.
Recorded temperatures:
Time: 2.0 hours, Temperature: 9.0°C
Time: 3.0 hours, Temperature: 6.0°C
Time: 4.0 hours, Temperature: 12.0°C
Predict the temperature: 3.5 hours?

Activity 1:

Identify attributes suitable for applying Linear regression. Construct a linear regression model for your dataset and predict the missing values in your data set. Evaluate the accuracy of prediction.(usage of built in package for prediction is not expected)

Activity 2:

Identify attributes suitable for applying Multiple Linear regression. Construct a linear regression model for your dataset and predict the missing values in your data set. Evaluate the accuracy of prediction.(usage of built in package for prediction is not expected)

Code with Output/Result:

Activity 0:

```
import numpy as np
# Sample data points
known_x = [1, 2, 3, 4, 5]
known_y = [5, 9, 6, 12, 8]
# New data point to predict
new_x = 3.5
# Perform linear interpolation
interp_value = np.interp(new_x, known_x, known_y)
print(f"Interpolated value at x = {new_x}: {interp_value}")
```

```
Interpolated value at x = 3.5: 9.0
```

Activity 1:

Single Linear Regression:

```
import numpy as np
age=np.array([6,8,7,5,10,9,3]).reshape((-1,1))
height=np.array([3.2,4.1,3.6,3.0,4.2,4.1,2.7])
print(age)
print(height)
```

```
[[ 6]
 [ 8]
 [ 7]
 [ 5]
 [10]
 [ 9]
 [ 3]]
[3.2 4.1 3.6 3.  4.2 4.1 2.7]
```

```python
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(age,height)
model=LinearRegression().fit(age,height)
rsq=model.score(age,height)
print("Co-Efficient of determination: ",rsq)
```

```
Co-Efficient of determination:  0.942472354890065
```

```python
print("Intercept: ",model.intercept_)
print("Slope: ",model.coef_)
```

```
Intercept:  1.8934426229508206
Slope:  [0.24262295]
```

```python
ypre=model.intercept_+model.coef_*age
print("Predicate Response: ",ypre, sep='\n')
```

```
Predicate Response:
[[3.34918033]
 [3.83442623]
 [3.59180328]
 [3.10655738]
 [4.31967213]
 [4.07704918]
 [2.62131148]]
```

Single Linear Regression without readymade functions:

```
import pandas as pd
age_height={
    "Age":[6,8,7,5,10,9,3],
    "Height":[3.2,4.1,3.6,3.0,4.2,4.1,2.7]
}
df=pd.DataFrame(age_height)
sum_x=df["Age"].sum()
sum_y=df["Height"].sum()
sum_x2=(df["Age"]**2).sum()
sum_xy=(df["Age"]*df["Height"]).sum()
intercept=(sum_y*sum_x2 - sum_x*sum_xy)/(len(df)*sum_x2 - sum_x**2)
slope=(len(df)*sum_xy - sum_x*sum_y)/(len(df)*sum_x2 - sum_x**2)
print("Intercept: ",intercept,"\nSlope: ",slope)
```

```
Intercept:  1.8934426229508272
Slope:  0.2426229508196714
```

```
y=slope*(4) + intercept
print("Predicted Value: ",y)
```

```
Predicted Value:  2.8639344262295126
```

Activity 2:

Multiple Linear Regression:

```
import numpy as np
import pandas as pd
week_age_height={
    "Week":[37,39,38,35,38,39,36,37],
    "Age":[6,8,7,5,10,9,3,4],
    "Height":[3.2,4.1,3.6,3.0,4.2,4.1,2.7,2.86]
}
df=pd.DataFrame(week_age_height)
print(df)
```

```
   Week  Age  Height
0    37    6    3.20
1    39    8    4.10
2    38    7    3.60
3    35    5    3.00
4    38   10    4.20
5    39    9    4.10
6    36    3    2.70
7    37    4    2.86
```

```
from sklearn.linear_model import LinearRegression
X=df[["Week","Age"]]
y=df["Height"]
model=LinearRegression().fit(X.values,y)
print("Multiple Linear Regression Predicted: ",model.predict([[38,11]]))
```

```
Multiple Linear Regression Predicted:  [4.42133056]
```

```
score = model.score(X,y)
print(score)
```

```
0.9757971691571447
```

```
model.intercept_
```

```
-1.7075675675675628
```

```
model.coef_
```

```
array([0.10428274, 0.19692308])
```

Multiple Linear Regression without readymade functions:

```
sum_week=df["Week"].sum()
sum_week2=(df["Week"]**2).sum()
sum_age=df["Age"].sum()
sum_age2=(df["Age"]**2).sum()
sum_height=df["Height"].sum()
sum_ageheight=(df["Age"]*df["Height"]).sum()
sum_weekheight=(df["Week"]*df["Height"]).sum()
sum_weekage=(df["Week"]*df["Age"]).sum()
sum_week2 = sum_week2 - sum_week**2/len(df)
sum_age2 = sum_age2 - sum_age**2/len(df)
sum_weekheight = sum_weekheight - (sum_week*sum_height)/len(df)
sum_ageheight = sum_ageheight - (sum_age*sum_height)/len(df)
sum_weekage = sum_weekage - (sum_week*sum_age)/len(df)
```

```
b1=(sum_age2*sum_weekheight - sum_weekage*sum_ageheight)/(sum_age2*sum_week2 - (sum_weekage)**2)
b2=(sum_week2*sum_ageheight - sum_weekage*sum_weekheight)/(sum_age2*sum_week2 - (sum_weekage)**2)
b0=df["Height"].mean() - b1*df["Week"].mean() - b2*df["Age"].mean()
```

```
Y=b0+ b1*38 + b2*11
print(Y)
```

```
4.421330561330578
```

```
import numpy as np
import pandas as pd
data=pd.read_csv(r'C:\Users\daxay\Downloads\Flight_delay.csv')
data_array=data.to_numpy()
print("Dataframe:\n",data_array)
arrdelay=data_array[:,12]
depdelay=data_array[:,13]
```

```
print("\nArrival Delay:\n",arrdelay)
print("\nDepature Delay:\n",depdelay)
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(arrdelay.reshape(-1, 1),depdelay.reshape(-1, 1))
rsq=model.score(arrdelay.reshape(-1, 1),depdelay.reshape(-1, 1))
print("\nCo-Efficient of determination: ",rsq)
print("\nIntercept: ",model.intercept_)
print("\nSlope: ",model.coef_)
ypre=model.predict(arrdelay.reshape(-1,1))
print("\nPredicate Response: ",ypre, sep='\n')
```

```
Dataframe:
 [[4 '03-01-2019' 1829 ... 0 0 32]
 [4 '03-01-2019' 1937 ... 0 0 47]
 [4 '03-01-2019' 1644 ... 0 0 72]
 ...
 [2 '17-06-2019' 1617 ... 5 0 20]
 [7 '22-06-2019' 1607 ... 0 0 25]
 [1 '23-06-2019' 1608 ... 0 0 0]]

Arrival Delay:
 [34 57 80 ... 47 26 18]

Depature Delay:
 [34 67 94 ... 42 32 33]

Co-Efficient of determination:  0.9003283583894522

Intercept:  [0.7038967]

Slope:  [[0.93246222]]

Predicate Response:
[[32.40761231]
 [53.85424346]
 [75.30087461]
 ...
 [44.52962122]
 [24.94791452]
 [17.48821673]]
```

```
import numpy as np
import pandas as pd
data=pd.read_csv(r'C:\Users\daxay\Downloads\Flight_delay.csv')
data_array=data.to_numpy()
print("Data:\n",data_array)
x={
    "airtime":data_array[:,11],
```

```
    "arrdelay":data_array[:,12],
    "depdelay":data_array[:,13]
}
df=pd.DataFrame(x)
print("\nDataframe:\n",df)
from sklearn.linear_model import LinearRegression
X=df[["arrdelay","depdelay"]]
y=df["airtime"]
model=LinearRegression().fit(X.values,y)
print("\nMultiple Linear Regression Predicted: ",model.predict([[38,11]]))
score = model.score(X,y)
print("\nScore: ",score)
print("\nIntercept: ",model.intercept_)
print("\nSlope: ",model.coef_)
```

```
Data:
 [[4 '03-01-2019' 1829 ... 0 0 32]
 [4 '03-01-2019' 1937 ... 0 0 47]
 [4 '03-01-2019' 1644 ... 0 0 72]
 ...
 [2 '17-06-2019' 1617 ... 5 0 20]
 [7 '22-06-2019' 1607 ... 0 0 25]
 [1 '23-06-2019' 1608 ... 0 0 0]]

Dataframe:
         airtime  arrdelay  depdelay
0            77        34        34
1           230        57        67
2           107        80        94
3           213        15        27
4           110        16        28
...         ...       ...       ...
484546      131        27        34
484547      136        39        41
484548      141        47        42
484549      137        26        32
484550      129        18        33

[484551 rows x 3 columns]

Multiple Linear Regression Predicted:  [116.21989906]
```

```
Score:  0.008812339503334266

Intercept:  106.11901217546037

Slope:  [ 0.36334221 -0.33691974]
```

Post Lab Question-Answers:

1. How will you choose between linear regression and non-linear regression?

Ans: When choosing between linear regression and non-linear regression, you need to consider the nature of your data and the relationship between the variables you are analyzing. Here are some factors to consider:

1. Linearity of the relationship: If the relationship between the independent and dependent variables appears to be linear, with a straight-line pattern, then linear regression is appropriate. On the other hand, if the relationship seems to follow a curved or non-linear pattern, non-linear regression may be more suitable.

2. Complexity of the model: Linear regression models are simpler and easier to interpret since they assume a linear relationship between variables. Non-linear regression models can capture more complex relationships but may be more challenging to interpret and require more computational resources.

3. Domain knowledge: Consider your understanding of the underlying process or theory related to the data. If you have prior knowledge suggesting a specific non-linear relationship, it may be more appropriate to use non-linear regression.

4. Data availability: Non-linear regression models often require more data points to estimate the parameters accurately. If you have a limited dataset, linear regression may be a more viable option.

5. Model performance: Evaluate the performance of both linear and non-linear regression models using appropriate metrics (e.g., R-squared, mean squared error). Compare their predictive accuracy and choose the model that performs better on your data.

2. Explain the nature or characteristics of a dataset where we can apply regression imputation.

Ans: Regression imputation is a technique used to fill in missing values in a dataset by predicting them based on the relationship between the variables. It assumes that the missing values are related to other variables in the dataset and can be estimated using regression analysis.

The characteristics of a dataset where regression imputation can be applied are as follows:

1. Missing at random (MAR): Regression imputation assumes that the missing values are not systematically related to the missing values themselves. In other words, the missingness is related to other observed variables in the dataset. If the missingness is random or can be explained by other variables, regression imputation can be effective.

2. Linear relationship: Regression imputation assumes a linear relationship between the variable with missing values and the other variables used for prediction. If the relationship is non-linear, other techniques like non-linear regression imputation or machine learning algorithms may be more appropriate.

3. Sufficient predictor variables: Regression imputation requires having other variables in the dataset that are strongly correlated with the variable with missing values. These predictor variables should be able to explain a significant portion of the variation in the variable with missing values.

4. Independence of errors: Regression imputation assumes that the errors in the regression model are independent and normally distributed. This assumption ensures that the imputed values are unbiased and have reasonable variability.

5. Adequate sample size: Regression imputation performs better with larger sample sizes since it relies on estimating the regression coefficients accurately. With a small sample size, the imputed values may be less reliable.

Outcomes:
Comprehend descriptive and proximity measures of data

Conclusion (based on the Results and outcomes achieved):
The experiment demonstrated the potential of regression modeling for predicting missing data values, providing researchers with a valuable tool for data imputation and analysis. Further research and experimentation can explore the limitations and applicability of regression imputation in different contexts and datasets.

References:

Books/ Journals/ Websites

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition

---