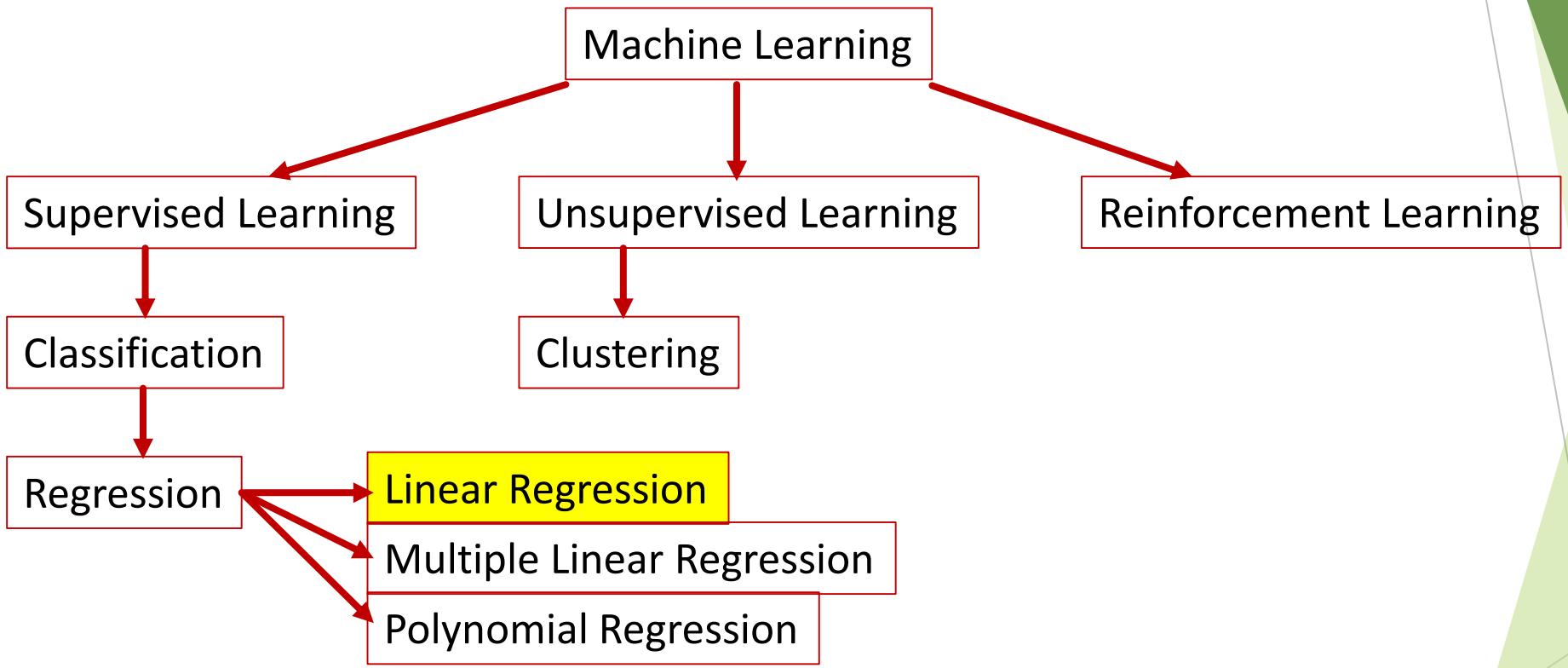
Certification Program in Business Analytics & Optimisation From IITD

Assignment 3
Linear Regression using PYTHON

From:
Shivesh Kumar Sareen
shivesh72@gmail.com

Objective:
Linear Regression using Python from Module 4



In this assignment, we will focus on Linear Regression aspect of Supervised algorithm using Python.

Steps for Linear Regression using Python

Step 1: Importing Libraries : Certain libraries like Numpy, Pandas and Matplotlib needs to be imported to carry out data pre-processing.

Step 2: Loading the dataset : Dataset which is a (.csv) file needs to be loaded in Python.

Step 3: Defining the variables : Input and output parameters need to be defined.

Step 4: Handling the missing values: Dataset contains empty or missing cells. These values need to be handled.

Step 5: Splitting the dataset : To develop a machine learning model, dataset needs to be divided into training and test data. Bigger the dataset, more is the training data, better is the accuracy of the model.

Step 6: Training the Simple Linear Regression model on the Training set

Step 7: Predicting the test set results

Step 8: Visualising the training set results

Step 9: Visualising the test set results

Sample code where Linear Regression has been done

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Libraries are imported in Python

```
# Importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
```

Datasets which are to be studied are imported in Python

```
# Defining the variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(X)
print(y)
```

```
# Taking care of missing data
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
print(X)
```

Missing data is replaced with the mean of other data.

```
# 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 = 1)
print(X_train)
print(X_test)
print(y_train)
print(y_test)
```

80% of the data is used for training the model and 20% is used for testing the model, to check the accuracy

Name	Type	Size	Value
dataset	DataFrame	(30, 2)	Column names: YearsExperience, Salary
regressor	linear_model._base.LinearRegression	1	LinearRegression object of sklearn.linear_model._base modu
X	Array of float64	(30, 1)	[[1.1] [1.3]
X_test	Array of float64	(10, 1)	[[1.5] [10.3]
X_train	Array of float64	(20, 1)	[[2.9] [5.1]
y	Array of float64	(30,)	[39343. 46205. 37731. ... 112635. 122391. 121872.]
y_pred	Array of float64	(10,)	[40835.10590871 123079.39940819 65134.55626083 63265.36 1156 ...
y_test	Array of float64	(10,)	[37731. 122391. 57081. 63218. 116969. 109431. 112635. ...
y_train	Array of float64	(20,)	[56642. 66029. 64445. ... 98273. 67938. 56957.]

```
# Training the Simple Linear Regression model on 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)
```

```
# Visualising the Training set results
plt.scatter(X_train, y_train, color = 'orange')
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 (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

THANK YOU