==========

In the Name of Allah, the Most Beneficent, the Most Merciful

==========

```
*----------------------------- AUTHOR_DETAILS ------------------------------*
|                                                                           |
|        Project Title  = GPA Prediction System                             |
|                                                                           |
|        Author         = Mr. Atif Raza Chaudary                            |
|                                                                           |
|        Copyright      = Copyright (C) 2020 Mr. Atif Raza Chaudary          |
|                                                                           |
|        License        = Public Domain                                     |
|                                                                           |
|        Version        = 1.0                                               |
|                                                                           |
*---------------------------------------------------------------------------*
```

---------- PROJECT PURPOSE ----------

The main purpose of this Project is to demonstrate how the GPA Prediction Problem can be treated as a Supervised Machine Learning Problem using Python and Scikit-learn Machine Learning Toolkit

For this Purpose, In Sha Allah, we will execute the Machine Learning Cycle

---

GPA Prediction System – Machine Learning Cycle

Machine Learning Cycle

Four phases of a Machine Learning Cycle are

Training Phase

```
Build the Model using Training Data
```

Testing Phase

```
Evaluate the performance of Model using Testing Data
```

Application Phase

```
Deploy the Model in the Real-world, to predict Real-time unseen Data
```

Feedback Phase

```
Take Feedback from the Users and Domain Experts to improve the Model
```

Executing Machine Learning Cycle Using a Single File

In Sha Allah, we will follow the following Steps to execute the Machine Learning Cycle Using a Single File

Step 1: Import Libraries

Step 2: Load Sample Data

Step 3: Understand and Pre-process Sample Data

Step 3.1: Understand Sample Data

Step 3.2: Pre-process Sample Data

Step 4: Feature Extraction

Step 5: Label Encoding (Input and Output is converted in Numeric Representation)

Step 5.1: Train the Label Encoder

Step 5.2: Label Encode the Output

Step 5.3: Label Encode the Input

Step 6: Execute the Training Phase

Step 6.1: Splitting Sample Data into Training Data and Testing Data

Step 6.2: Splitting Input Vectors and Outputs/Labels of Training Data

Step 6.3: Train the Support Vector Regressor

Step 6.4: Save the Trained Model

Step 7: Execute the Testing Phase

```
Step 7.1: Splitting Input Vectors and Output/Labels of Testing Data

Step 7.2: Load the Saved Model

Step 7.3: Evaluate the Performance of Trained Model

    Step 7.3.1: Make Predictions from the Model on Testing Data

Step 7.4: Calculate the Mean Absolute Error.
```

Step 8: Execute the Application Phase

```
Step 8.1: Take Input from User

Step 8.2: Convert User Input into Feature Vector (Exactly Same as Feature Vectors of Sample Data)

Step 8.3: Label Encoding of Feature Vector (Exactly Same as Label Encoded Feature Vectors of Sample Dat
a)

Step 8.4: Load the Saved Model

Step 8.5: Model Prediction

    Step 8.5.1: Apply Model on the Label Encoded Feature Vector of unseen instance and return Predictio
n to the User
```

Step 9: Execute the Feedback Phase

Step 10: Improve the Model based on Feedback

Step 1: Import Libraries

In [1]:
```
# Import Libraries

import re
import scipy
import pickle
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVR
from sklearn.metrics import mean_absolute_error
from prettytable import PrettyTable
from astropy.table import Table, Column
```

Step 2: Load Sample Data

In [2]:
```
# Load Sample Data

'''
*---------------------- LOAD_SAMPLE_DATA ------------------------*
|     Function: read_csv()                                      |
|             Purpose: Read a dataset in CSV file format        |
|     Arguments:                                                |
|             path: Path to dataset file                        |
|             dataset: Dataset file name                        |
|     Return:                                                   |
|             dataset: Dataset in DataFrame format              |
*---------------------------------------------------------------*
'''

sample_data = pd.read_csv("sample-data.csv")

print("\n\nSample Data:")
print("============\n")
pd.set_option("display.max_rows", None, "display.max_columns", None)
print(sample_data)
```

```
Sample Data:
============

    Matric Marks  FSc Marks   GPA
0            852        783  3.07
1            808        801  3.36
2            840        806  3.77
3            824        720  2.98
4            902        789  3.16
5            926        791  3.23
6            770        658  2.97
7            690        626  2.72
8            729        713  3.26
9            781        597  3.30
10           591        692  2.73
11           806        844  3.43
12           818        720  3.21
```

Step 3: Understand and Pre-process Sample Data

Step 3.1: Understand Sample Data

In [3]:
```
# Understand Sample Data

print("\n\nAttributes in Sample Data:")
print("==========================\n")

print(sample_data.columns)

print("\n\nNumber of Instances in Sample Data:",sample_data["Matric Marks"].count())
print("=======================================\n")
```

```
Attributes in Sample Data:
==========================

Index(['Matric Marks', 'FSc Marks', 'GPA'], dtype='object')


Number of Instances in Sample Data: 100
=======================================
```

Step 3.2: Pre-process Sample Data

- Sample Data is already Preprocessed
- No Preprocessing needs to be Performed

Step 4: Feature Extraction

- Features are already Extracted
- No Feature Extraction needs to be Performed

Step 5: Label Encoding the Sample Data (Input and Output is converted in Numeric Representation)

Step 5.1: Train the Label Encoder

- As Sample Data is already in Numeric Representation.
- Therefore, we will not Label Encode the Sample Data.

Step 5.2: Label Encode the Output

- As Output (GPA Attribute) is already in Numeric Representation.
- Therefore, we will not Label Encode the Output.

Step 5.3: Label Encode the Input

- As Input (Matric Marks and FSc Marks Attributes) is already in Numeric Representation.
- Therefore, we will not Label Encode the Input.

Step 6: Execute the Training Phase

Step 6.1: Splitting Sample Data into Training Data and Testing Data

In [4]:
```
# Splitting Sample Data into Training Data and Testing Data

'''
*------------------- SPLIT_SAMPLE_DATA ---------------------*
|        Function: train_test_split()                       |
|              Purpose: Split arrays or matrices into       |
|                       random train and test subsets       |
|        Arguments:                                         |
|              arrays: sequence of indexables               |
|              test_size: float or int                      |
|        Return:                                            |
|              splitting: list                              |
*-----------------------------------------------------------*
'''

training_data, testing_data = train_test_split( sample_data , test_size=0.2 , random_state=0 , shuffle = False)

# Save the Training and Testing Data into CSV File

training_data.to_csv(r'training-data.csv', index = False, header = True)
testing_data.to_csv(r'testing-data.csv', index = False, header = True)

# print Training and Testing Data

print("\n\nTraining Data:")
print("==============\n")
pd.set_option("display.max_rows", None, "display.max_columns", None)
print(training_data)
print("\n\nTesting Data:")
print("==============\n")
pd.set_option("display.max_rows", None, "display.max_columns", None)
print(testing_data)
```

```
Training Data:
==============

    Matric Marks  FSc Marks   GPA
0            852        783  3.07
1            808        801  3.36
2            840        806  3.77
```

3	824	720	2.98
4	902	789	3.16
5	926	791	3.23
6	770	658	2.97
7	690	626	2.72
8	729	713	3.26
9	781	597	3.30
10	591	692	2.73
11	806	844	3.43
12	818	720	3.21
13	828	748	2.84
14	770	698	2.64
15	594	715	3.47
16	871	789	3.35
17	785	718	3.29
18	854	752	3.54
19	859	673	2.95
20	790	769	3.13
21	800	665	2.96
22	906	764	3.00
23	693	690	3.07
24	745	667	3.33
25	717	720	3.26
26	696	725	2.45
27	697	729	3.34
28	867	735	3.05
29	831	723	2.94
30	732	761	3.38
31	802	686	3.38
32	715	688	2.75
33	745	642	2.81
34	758	851	3.57
35	764	735	3.12
36	822	674	2.90
37	855	839	3.58
38	886	821	3.56
39	538	615	2.96
40	954	866	3.59
41	764	677	2.30
42	893	721	2.87
43	749	723	2.87
44	798	764	3.45
45	729	779	2.84

	Matric Marks	FSc Marks	GPA
46	714	700	3.20
47	822	683	3.36
48	855	711	2.95
49	803	761	3.07
50	718	688	2.73
51	679	702	3.26
52	850	833	3.31
53	622	720	3.11
54	803	650	3.35
55	734	800	2.96
56	725	792	3.26
57	611	685	2.55
58	692	617	1.33
59	693	712	3.31
60	641	740	3.01
61	734	706	3.48
62	700	775	2.81
63	756	761	3.53
64	739	685	2.60
65	764	608	2.89
66	794	694	2.86
67	846	766	3.56
68	632	702	2.25
69	858	582	2.63
70	852	632	3.26
71	526	630	2.55
72	811	586	2.40
73	748	674	2.46
74	688	624	2.81
75	849	810	3.40
76	881	802	3.56
77	660	552	2.88
78	758	714	2.83
79	850	768	2.85

Testing Data:

==========

	Matric Marks	FSc Marks	GPA
80	578	648	2.71
81	905	762	2.88
82	806	684	2.63

```
83          686         798  2.63
84          784         676  2.88
85          729         716  3.01
86          826         750  2.60
87          622         616  2.41
88          744         610  2.88
89          895         646  2.83
90          662         676  3.08
91          743         756  2.88
92          814         764  2.85
93          686         548  2.68
94          796         598  2.53
95          783         650  2.93
96          817         668  2.98
97          803         650  3.35
98          734         800  2.96
99          725         792  3.26
```

6.2: Splitting Input Vectors and Outputs / Labels of Training Data

In [5]:
```
# Splitting Input Vectors and Outputs / Labels of Training Data

'''
*---------------- SPLIT_INPUT_VECTORS_AND_LABELS --------------*
|        Function: iloc()                                      |
|            Purpose: Splitting Input Vector and Labels        |
|        Arguments:                                            |
|            Attribute: Name or Location Attribute to Split    |
|        Return:                                               |
|            Attribute: Split Attributes                       |
*--------------------------------------------------------------*
'''
print("\n\nInputs Vectors (Feature Vectors) of Training Data:")
print("================================================\n")
input_vector_train = training_data.iloc[:, 0:2]
print(input_vector_train)

print("\n\nOutputs/Labels of Training Data:")
print("================================\n")
print("  GPA")
output_label_train = training_data.iloc[:,2:]
print(output_label_train)
```

```
Inputs Vectors (Feature Vectors) of Training Data:
================================================

    Matric Marks  FSc Marks
0            852        783
1            808        801
2            840        806
3            824        720
4            902        789
5            926        791
6            770        658
7            690        626
8            729        713
9            781        597
10           591        692
11           806        844
12           818        720
```

13	828	748
14	770	698
15	594	715
16	871	789
17	785	718
18	854	752
19	859	673
20	790	769
21	800	665
22	906	764
23	693	690
24	745	667
25	717	720
26	696	725
27	697	729
28	867	735
29	831	723
30	732	761
31	802	686
32	715	688
33	745	642
34	758	851
35	764	735
36	822	674
37	855	839
38	886	821
39	538	615
40	954	866
41	764	677
42	893	721
43	749	723
44	798	764
45	729	779
46	714	700
47	822	683
48	855	711
49	803	761
50	718	688
51	679	702
52	850	833
53	622	720
54	803	650
55	734	800

```
56          725         792
57          611         685
58          692         617
59          693         712
60          641         740
61          734         706
62          700         775
63          756         761
64          739         685
65          764         608
66          794         694
67          846         766
68          632         702
69          858         582
70          852         632
71          526         630
72          811         586
73          748         674
74          688         624
75          849         810
76          881         802
77          660         552
78          758         714
79          850         768


Outputs/Labels of Training Data:
================================

  GPA
     GPA
0   3.07
1   3.36
2   3.77
3   2.98
4   3.16
5   3.23
6   2.97
7   2.72
8   3.26
9   3.30
10  2.73
11  3.43
```

```
12  3.21
13  2.84
14  2.64
15  3.47
16  3.35
17  3.29
18  3.54
19  2.95
20  3.13
21  2.96
22  3.00
23  3.07
24  3.33
25  3.26
26  2.45
27  3.34
28  3.05
29  2.94
30  3.38
31  3.38
32  2.75
33  2.81
34  3.57
35  3.12
36  2.90
37  3.58
38  3.56
39  2.96
40  3.59
41  2.30
42  2.87
43  2.87
44  3.45
45  2.84
46  3.20
47  3.36
48  2.95
49  3.07
50  2.73
51  3.26
52  3.31
53  3.11
54  3.35
```

```
55  2.96
56  3.26
57  2.55
58  1.33
59  3.31
60  3.01
61  3.48
62  2.81
63  3.53
64  2.60
65  2.89
66  2.86
67  3.56
68  2.25
69  2.63
70  3.26
71  2.55
72  2.40
73  2.46
74  2.81
75  3.40
76  3.56
77  2.88
78  2.83
79  2.85
```

6.3: Train the Support Vector Regressor

In [6]:
```
# Train the Support Vector Regressor

'''
*--------------- TRAIN_SUPPORT_VECTOR_REGRESSOR  ------------------*
|       Function: svm.LinearSVR()                                  |
|           Purpose: Train the Algorithm on Training Data          |
|       Arguments:                                                 |
|           Training Data: Provide Training Data to the Model      |
|       Return:                                                    |
|           Parameter: Model return the Training Parameters        |
*------------------------------------------------------------------*
'''
print("\n\nTraining the Support Vector Regressor on Training Data")
print("=======================================================\n")
print("\nParameters and their values:")
print("============================\n")
svr_model = LinearSVR()
svr_model.fit(input_vector_train,np.ravel(output_label_train))
print(svr_model)
```

```
Training the Support Vector Regressor on Training Data
=======================================================


Parameters and their values:
============================

LinearSVR(C=1.0, dual=True, epsilon=0.0, fit_intercept=True,
     intercept_scaling=1.0, loss='epsilon_insensitive', max_iter=1000,
     random_state=None, tol=0.0001, verbose=0)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:931: ConvergenceWarning: Liblinear failed to co
nverge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Step 6.4: Save the Trained Models

In [7]:
```
# Save the Trained Models

'''
*--------------------- SAVE_THE_TRAINED_MODELS --------------------*
|        Function: dump()                                         |
|             Purpose: Save the Trained Model on your Hard Disk   |
|        Arguments:                                               |
|             Model: Model Objects                                |
|        Return:                                                  |
|             File: Trained Model will be Saved on Hard Disk      |
*-----------------------------------------------------------------*
'''

# Save the Models in a Pkl File

pickle.dump(svr_model, open('svr_trained_model.pkl', 'wb'))
```

Step 7: Execute the Testing Phase

Step 7.1: Splitting Input Vectors and Outputs / Labels of Testing Data

In [8]:
```
# Splitting Input Vectors and Outputs/Labels of Testing Data

'''
*---------------- SPLIT_INPUT_VECTORS_AND_LABELS --------------*
|        Function: iloc()                                      |
|            Purpose: Splitting Input Vector and Labels        |
|        Arguments:                                            |
|            Attribute: Name or Location Attribute to Split    |
|        Return:                                               |
|            Attribute: Split Attributes                       |
*--------------------------------------------------------------*
'''

print("\n\nInputs Vectors (Feature Vectors) of Testing Data:")
print("================================================\n")
input_vector_test = testing_data.iloc[:, 0:2]
print(input_vector_test)

print("\n\nOutputs/Labels of Testing Data:")
print("==============================\n")
print("  GPA")
output_label_test = testing_data.iloc[:, 2:]
print(output_label_test)
```

```
Inputs Vectors (Feature Vectors) of Testing Data:
================================================

    Matric Marks  FSc Marks
80           578        648
81           905        762
82           806        684
83           686        798
84           784        676
85           729        716
86           826        750
87           622        616
88           744        610
89           895        646
90           662        676
91           743        756
```

```
92          814        764
93          686        548
94          796        598
95          783        650
96          817        668
97          803        650
98          734        800
99          725        792
```

```
Outputs/Labels of Testing Data:
==============================
```

```
  GPA
     GPA
80  2.71
81  2.88
82  2.63
83  2.63
84  2.88
85  3.01
86  2.60
87  2.41
88  2.88
89  2.83
90  3.08
91  2.88
92  2.85
93  2.68
94  2.53
95  2.93
96  2.98
97  3.35
98  2.96
99  3.26
```

Step 7.2: Load the Saved Model

In [9]:
```
# Load the Saved Model

'''
*------------------- LOAD_SAVED_MODEL --------------------------*
|         Function: load()                                     |
|               Purpose: Method to Load Previously Saved Model |
|         Arguments:                                           |
|               Model: Trained Model                           |
|         Return:                                              |
|               File: Saved Model will be Loaded in Memory     |
*--------------------------------------------------------------*
'''

# Load the Saved Model

model = pickle.load(open('svr_trained_model.pkl', 'rb'))
```

Step 7.3: Evaluate the Machine Learning Model

Step 7.3.1: Make Predictions with the Trained Models on Testing Data

In [10]:
```
# Evaluate the Machine Learning Model

'''
*--------------------- EVALUATE_MACHINE_LEARNING_MODEL ----------------------*
|       Function: Predict()                                                  |
|             Purpose: Make a Prediction using Algorithm on Test Data        |
|       Arguments:                                                           |
|            Testing Data: Provide Test data to the Trained Model            |
|       Return:                                                              |
|            Predictions: Model return Predictions                           |
*----------------------------------------------------------------------------*
'''

# Provide Test data to the Trained Model

model_predictions = model.predict(input_vector_test)
testing_data.copy(deep=True)
pd.options.mode.chained_assignment = None
testing_data["Predictions"] = np.round(model_predictions,2)

# Save the Predictions into CSV File

testing_data.to_csv(r'model-predictions.csv', index = False, header = True)

model_predictions = testing_data
print("\n\nPredictions Returned by svr_trained_model:")
print("=========================================\n")
print(model_predictions)
```

```
Predictions Returned by svr_trained_model:
==========================================

    Matric Marks  FSc Marks   GPA  Predictions
80           578        648  2.71         2.08
81           905        762  2.88         2.51
82           806        684  2.63         2.25
83           686        798  2.63         2.55
84           784        676  2.88         2.22
85           729        716  3.01         2.32
86           826        750  2.60         2.45
```

87	622	616	2.41	2.00
88	744	610	2.88	2.02
89	895	646	2.83	2.17
90	662	676	3.08	2.18
91	743	756	2.88	2.44
92	814	764	2.85	2.49
93	686	548	2.68	1.82
94	796	598	2.53	2.00
95	783	650	2.93	2.14
96	817	668	2.98	2.21
97	803	650	3.35	2.15
98	734	800	2.96	2.57
99	725	792	3.26	2.54

Step 7.4: Calculate the Mean Absolute Error

In [11]:
```
# Calculate the Root Mean Squared Error

'''
/*----------------------- CALCULATE_ROOT_MEAN_SQUARE_ERROR -----------------*
|          Function: mean_squared_error()                                  |
|                Purpose: Evaluate the algorithm on Testing data           |
|          Arguments:                                                      |
|                Prediction: Predicted values                              |
|                Label: Actual values                                      |
|          Return:                                                         |
|                Root Mean Squared Error                                   |
*--------------------------------------------------------------------------*
'''

# Calculate the Root Mean Squared Error

model_mbe = mean_absolute_error(model_predictions["GPA"],model_predictions["Predictions"])

print("\n\nMean Absolute Error:")
print("======================\n")
print(round(model_mbe,2))
```

```
Mean Absolute Error:
===============

0.59
```

Step 8: Execute the Application Phase

Step 8.1: Take Input from User

In [12]:
```
# Take Input from User

'''
*---------------- TAKE_USER_INPUT ----------------*
'''

matric_marks_input = input("\nPlease enter your Matric Marks : ").strip()
fsc_marks_input = input("\nPlease enter your FSc Marks : ").strip()
```

```
Please enter your Matric Marks : 871

Please enter your FSc Marks : 830
```

Step 8.2: Convert User Input into Feature Vector (Exactly Same as Feature Vectors of Sample Data)

In [13]:
```
# Convert User Input into Feature Vector

user_input = pd.DataFrame({ 'Matric Marks': [matric_marks_input],'FSC Marks': [fsc_marks_input]})

print("\n\nUser Input Feature Vector:")
print("==========================\n")
print(user_input)
```

```
User Input Feature Vector:
==========================

  Matric Marks FSC Marks
0          871       830
```

Step 8.3: Label Encoding of Feature Vector (Exactly Same as Label Encoded Feature Vectors of Sample Data)

- As Input of Unseen Instance (Matric Marks and FSc Marks Attributes) is already in Numeric Representation.
- Therefore, we will not Label Encode the Input of Unseen Instance.

Step 8.4: Load the Train Model

In [14]:
```
# Load the Saved Model

'''
*----------------------- LOAD_SAVED_MODEL --------------------------*
|         Function: load()                                         |
|             Purpose: Method to Load Previously Saved Model       |
|         Arguments:                                               |
|               Model: Trained Model                               |
|         Return:                                                  |
|               File: Saved Model will be Loaded in Memory         |
*------------------------------------------------------------------*
'''

# Load the Saved Model

model = pickle.load(open('svr_trained_model.pkl', 'rb'))
```

Step 8.5: Model Prediction

Step 8.5.1: Apply Model on the Label Encoded Feature Vector of unseen instance and return Prediction to the User

In [15]:
```
# Prediction of Unseen Instance

'''
*---------------------------  MODEL_PREDICTION --------------------------*
|           Function: predict()                                          |
|                 Purpose: Use Trained Model to Predict the Output       |
|                          of Unseen Instances                           |
|           Arguments:                                                   |
|                 User Data: Label Encoded Feature Vector of             |
|                            Unseen Instances                            |
|           Return:                                                      |
|                 GPA                                                    |
*------------------------------------------------------------------------*
'''

# Make a Prediction on Unseen Data

predicted_gpa = model.predict(user_input)

# Add the Prediction in a Pretty Table

pretty_table = PrettyTable()
pretty_table.add_column("       ** Prediction **       ",np.round(predicted_gpa,2))
print(pretty_table)
```

```
+--------------------------------+
|        ** Prediction **        |
+--------------------------------+
|              2.7               |
+--------------------------------+
```

Step 9: Execute the Feedback Phase

A Two-Step Process

Step 01: After some time, take Feedback from

- Domain Experts and Users on deployed GPA Prediction System

Step 02: Make a List of Possible Improvements based on Feedback received

Step 10: Improve Model based on Feedback

There is Always Room for Improvement

Based on Feedback from Domain Experts and Users

- Improve your Model

==========

JAZAK ALLAH KHAIR

==========