DeepSphere.AI
Enterprise AI and IIoT for Analytics

STUDENT GRADE

Learn how machine learning applied in grade prediction

DeepSphere. AI

Deepsphere.AI Is the Most Simple, Easy to Use, and Applied Artificial Intelligence Educational Platform. Our Platform Offers Two Unique Products to Reinvent Education to the Next Level AI Learning and Digital Education Transformation.

DeepSphere.AI Educational Platform

Powered By

Google Cloud — **AI Lab Infrstrucure**

iLMS and Industrial Curriculum — SAP

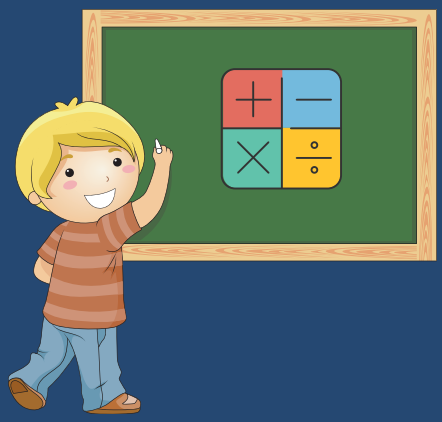
amazon web services — **Cloud Computing**

DeepSphere
Enterprise AI and IIoT for Ana

Contents

DeepSphere.AI
Enterprise AI and IIoT for Analytics

Disclaimer



All software and hardware used or referenced in this guide belong to their respective vendor. We developed this guide based on our development infrastructure, and this guide may or may not work on other systems and technical infrastructure. We are not liable for any direct or indirect problems caused by users using this guide.

Executive Summary

The purpose of this document is to provide adequate information to users to implement a machine learning model. To achieve this, we use one of the most common Student Grade prediction Problems.

DeepSphere.AI
Enterprise AI and IIoT for Analytics

Problem Statement

Grade prediction of 6th,7th and 8th Student in mathematics

DeepSphere.AI
Enterprise AI and IIoT for Analytics

Step-1 Working with Configuration File

The configuration file controls the machine learning model run time components such as data preprocessing, training, testing, and deploying the model. The configuration file format varies by the operating system, such as windows, Unix, and Linux.

```
import configparser
import os
vAR_read_config = configparser.ConfigParser(allow_no_value=True)
vAR_INI_FILE_PATH = os.environ.get('SCORE_PREDICTION_INI_FILE_PATH')
vAR_read_config.read(vAR_INI_FILE_PATH)
vAR_Training_Path = vAR_read_config['PATH']['Training_file']
vAR_Test_Path = vAR_read_config['PATH']['Test_file']
```

DeepSphere.AI
Enterprise AI and IIoT for Analytics

Step-2
Importing libraries

Configparser

The configparser module from Python's standard library defines functionality for reading and writing configuration files as used by Microsoft Windows OS. Such files usually have .INI extension.

Pandas

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure.

Sklearn

Sklearn is the Machine Learning Library which contains numerous other libraries like numpy, scipyetc. which are used for numerical & scientific computations.

```
import configparser
import pandas as vAR_pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_predict
import matplotlib.pyplot as vAR_plt
```


DeepSphere.AI
Enterprise AI and IIoT for Analytics

Step-3
Importing Training Data

Next immediate step after importing all libraries is getting the Training data imported. We are importing the Clustering data stored in our local system with the use of Pandas library.

```
vAR_df=vAR_pd.read_excel(vAR_Training_Path)
vAR_df
```

Step-3
Converting categorical to numerical

Converting all the categorical variable to numerical from the training data

DeepSphere.AI
Enterprise AI and IIoT for Analytics

```
obj_vAR_df=vAR_df.select_dtypes(include=['object']).copy()
obj_vAR_df
obj_vAR_df[obj_vAR_df.isnull().any(axis=1)]
obj_vAR_df["Final grade"] = obj_vAR_df["Final grade"].astype('category')
obj_vAR_df["Primary Language(Reading &writing)"]=obj_vAR_df["Primary Language(Reading &writing)"].astype('category')
obj_vAR_df["Secondary Language(Reading&writing)"]=obj_vAR_df["Secondary Language(Reading&writing)"].astype('category')
obj_vAR_df.dtypes
obj_vAR_df["Final grade_cat"] = obj_vAR_df["Final grade"].cat.codes
obj_vAR_df["Primary Language(Reading &writing)_cat"] = obj_vAR_df["Primary Language(Reading &writing)"].cat.codes
obj_vAR_df["Secondary Language(Reading&writing)_cat"] = obj_vAR_df["Secondary Language(Reading&writing)"].cat.codes
obj_vAR_df
obj_vAR_df["Final grade"].dtypes
obj_vAR_df["Primary Language(Reading &writing)"].dtypes
obj_vAR_df["Secondary Language(Reading&writing)"].dtypes
obj_vAR_df["Final grade_cat"].dtypes
obj_vAR_df["Primary Language(Reading &writing)_cat"].dtypes
obj_vAR_df["Secondary Language(Reading&writing)_cat"].dtypes
vAR_df = vAR_df.merge(obj_vAR_df["Final grade_cat"],left_index=True,right_index=True)
vAR_df = vAR_df.merge(obj_vAR_df["Primary Language(Reading &writing)_cat"],left_index=True,right_index=True)
vAR_df = vAR_df.merge(obj_vAR_df["Secondary Language(Reading&writing)_cat"],left_index=True,right_index=True)
vAR_df
```

Step-4
Selecting Features and Labels

```
vAR_Features_Train=vAR_df[['Assignment 1(%)','Assignment 2(%)','Assignment 3(%)','Semester 1(%)',
                           'Semester 2(%)','Backlogs','Tech-fest participation','Attendance',
                           'Quiz Voluntering','Maths Olympiad','Science olympiad',
                           'Secondary Language(Reading&writing)_cat','Adaptiveness','Social activity',
                           'Primary Language(Reading &writing)_cat']]
vAR_Label_Train=vAR_df['Final grade']
vAR_Features_Train
vAR_Label_Train
```

Here the selected features is Assignment 1(%), Assignment 2(%) , Assignment 3(%), Semester 1(%), Semester 2(%), Backlogs, Tech-fest participation, Attendance, Quiz Voluntering, Maths Olympiad, Science olympiad, Secondary Language(Reading&writing)_cat , Adaptiveness, Social activity and Primary Language(Reading &writing)_cat'

And the selected label is Final grade.

Step-5
Training model

Training the data makes the model to Learn, understand & recognize the Pattern in the data.
This is to fit the model on training.

```
In [28]: vAR_model = LinearRegression()
         vAR_model.fit(vAR_Features_Train,obj_vAR_df["Final grade_cat"])

Out[28]: LinearRegression()
```

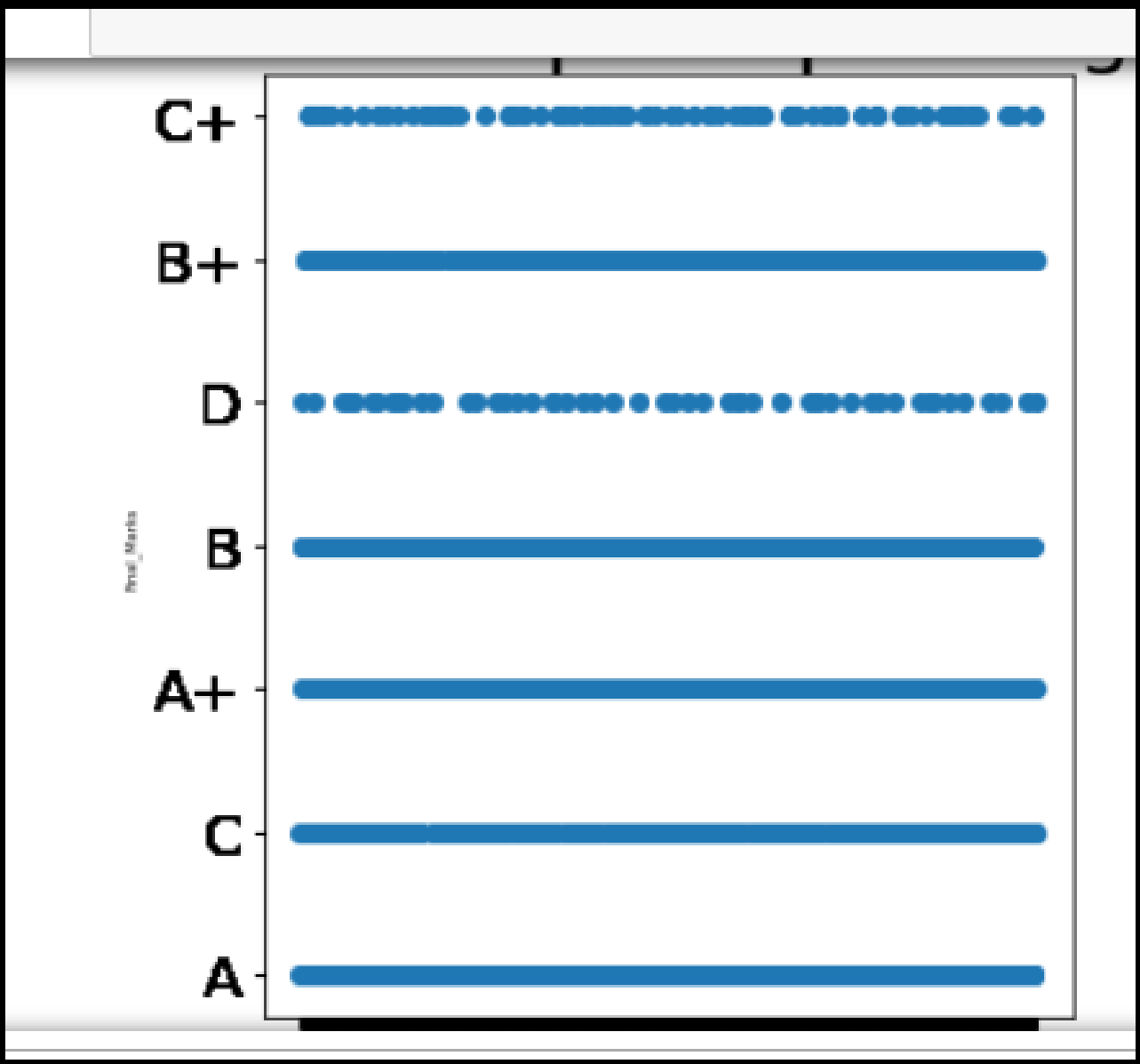
Some visualisation

```
x=vAR_df.iloc[:,0]
y=obj_vAR_df["Final grade"]
vAR_plt.scatter(x, y)
vAR_plt.xticks((x),size=20)
vAR_plt.yticks((y),size=20)
vAR_plt.title('Scatter point plotting',size=30)
vAR_plt.xlabel('id no.',size=6)

vAR_plt.ylabel('Final_Marks',size=5)

vAR_plt.show()
```

Plotting the grade of all the Students,where x-axis denotes the grade and y-axis denotes id no.



Gradient Descent

Gradient Descent is an algorithm that finds best fit line for given training data set with minimum overall. A loss is defined as a difference between predicted values and truth values .

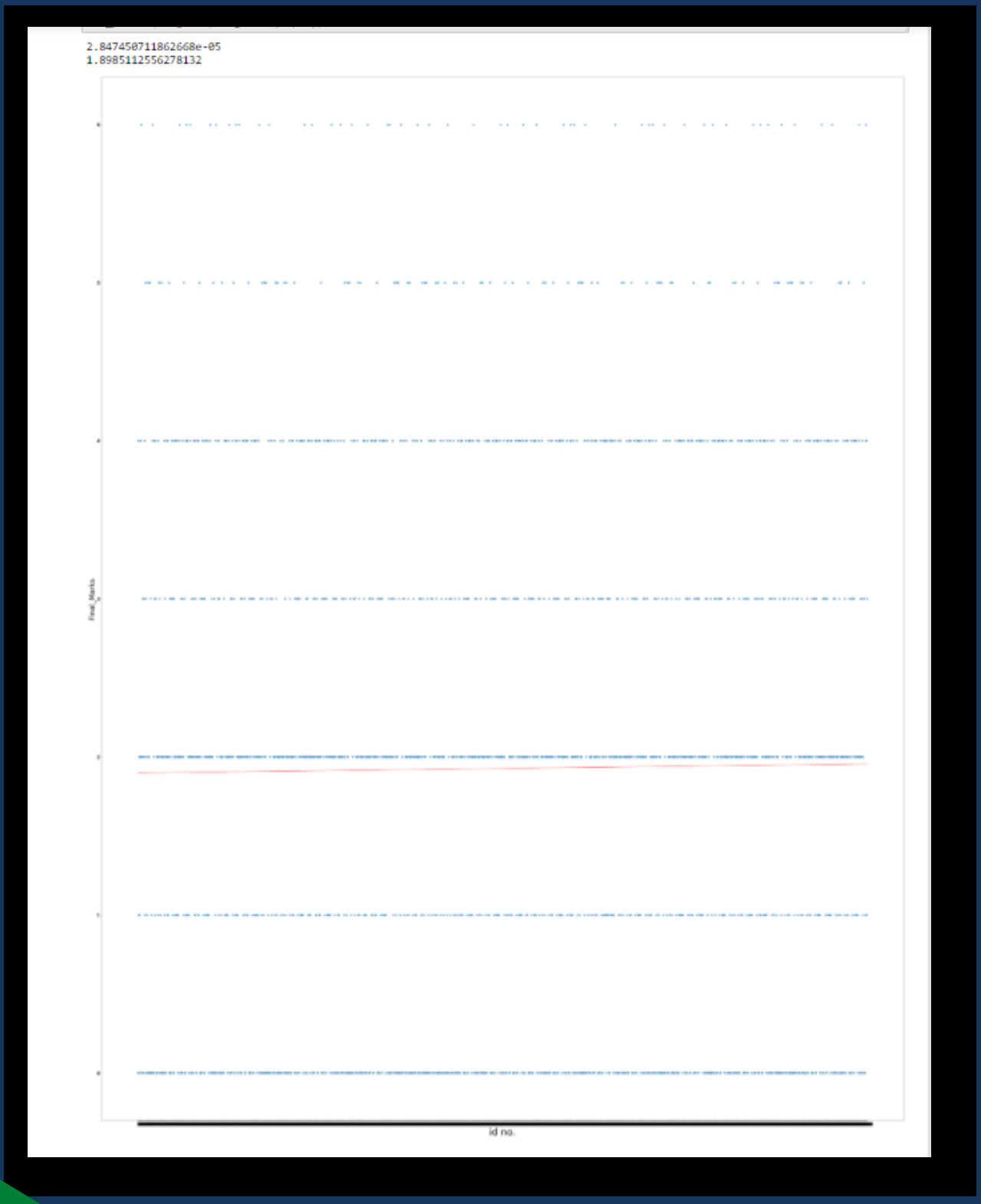
In [18]:
```
m = 0
b = 0
x=vAR_df.iloc[:,0]
y=obj_vAR_df["Final grade_cat"]
L = 0.01 # The learning Rate
epochs = 107  # The number of iterations to perform gradient descent

n = len(vAR_df) # Number of elements in training data
# Performing Gradient Descent
for i in range(epochs):
    Y_pred = m*x + b  # The current predicted value of Y
    D_m = (-2/n) * sum(x * (y - Y_pred))  # Derivative wrt m
    D_b = (-2/n) * sum(y - Y_pred)  # Derivative wrt b
    m = m - L * D_m  # Update m
    b = b - L * D_b  # Update b
```

In [19]:
```
x=vAR_df.iloc[:,0]
y=obj_vAR_df["Final grade_cat"]
vAR_plt.scatter(x, y)
vAR_plt.xlabel('id no.',size=40)
vAR_plt.ylabel('Final_Marks',size=30)
vAR_plt.rc('figure', figsize=(40,60))
vAR_plt.xticks((x),size=20)
vAR_plt.yticks((y),size=20)
m, b = np.polyfit(x, y,1)
print(m)
print(b)
vAR_plt.plot(x, m*x + b,"r")
vAR_plt.rc('figure', figsize=(60,80))
```

DeepSphere.AI
Enterprise AI and IIoT for Analytics

By Gradient Descent algorithm ,value of m(Slope) and b(intercept) comes as 2.847e(-05) and 1.898 respectively.
Red line indicates the best fit line.



Step-6
Reviewing Learning algorithm

This step help us to know how the model learn by the provided features.

```
In [32]: y_pred=vAR_model.predict(vAR_Features_Train)
```

Accuracy

```
]: accuracy=vAR_model.score(vAR_Features_Train,obj_vAR_df["Final grade_cat"])
   accuracy=accuracy*100
   accuracy
```

87.785%

Sum square error

Sum square error is the difference between Actual and predicted value. Here it represents the error for each student.

```
In [35]: #finding sum square error
         sse=(obj_vAR_df["Final grade_cat"]-y_pred)**2
         sse

Out[35]: 0       0.399025
         1       0.256623
         2       0.007546
         3       0.640882
         4       0.531585
                   ...
         1995    0.033821
         1996    0.330482
         1997    0.025914
         1998    0.138659
         1999    0.571911
         Name: Final grade_cat, Length: 2000, dtype: float64
```

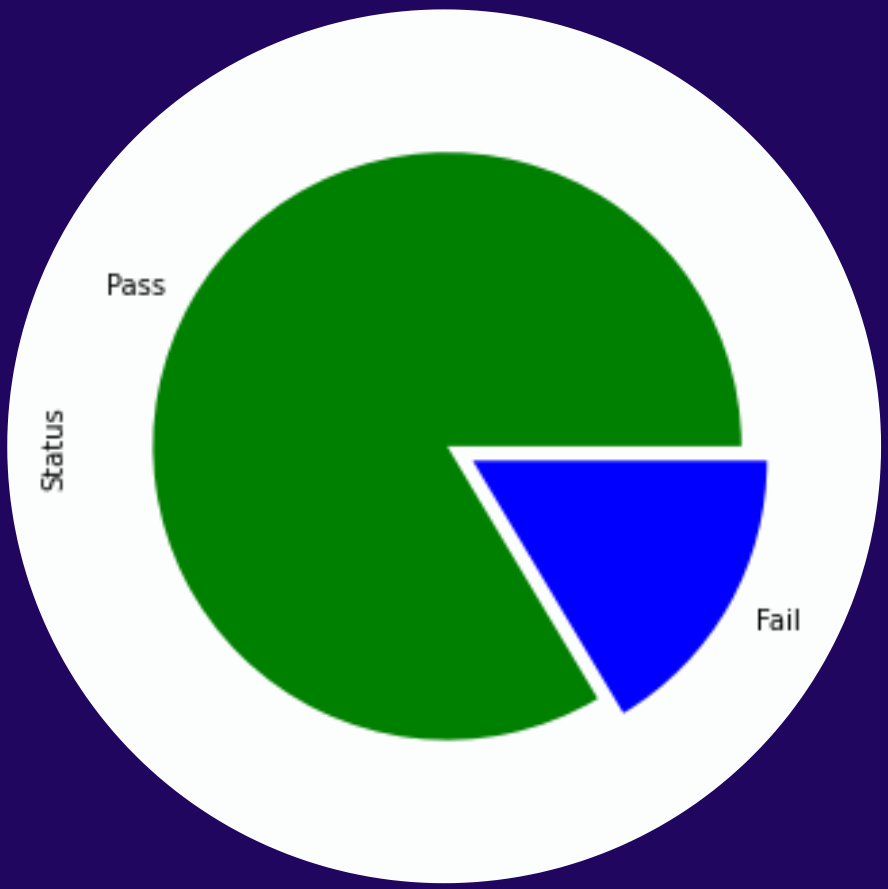
Mean square error

Mean square error is the average of sum square error. Here it represents the overall average error.

```
In [36]: mse=np.mean(sse)
         mse

Out[36]: 0.3394138558144636
```

Step-7
Status Visualisation




```
In [52]: data=obj_vAR_df["Status"].value_counts()
         colors = ['#7CB342','#FFB300']
         plot = data.plot.pie(y='Status',colors=colors, explode = [0,0.1], figsize=(5, 5))
```

Step-8
Different Validation Technique

K-Fold

K-fold validation is a method of validation which shuffles the data and splits it into k number of folds (groups). Value of k is to be choosed in such a way that each k fold(group) should be large enough to be representative of the model and small enough to be computed in a reasonable amount of time.

Loocv

Leave-one-out is a iterative validation technique where the number of folds equals the number of instances in the dataset. In a simple language ,if it has n number of data then (n-1) data is used for training and 1 data for testing. This process will be continue till each data goes for testing .

```
lm_k = vAR_model
k_predictions = cross_val_predict(lm_k, vAR_Features_Train, obj_vAR_df["Final grade_cat"], cv=10)
```

```
accuracy=vAR_model.score(vAR_Features_Train,k_predictions)
accuracy=accuracy*100
accuracy
```

```
99.99552627277691
```

```
LOO_predictions = cross_val_predict(lm_k,vAR_Features_Train , obj_vAR_df["Final grade_cat"], cv=(len(vAR_Features_Train)))
```

```
accuracy=vAR_model.score(vAR_Features_Train,LOO_predictions)
accuracy=accuracy*100
accuracy
```

```
99.99841245233063
```

DeepSphere.AI
Enterprise AI and IIoT for Analytics

70-30 Validation

In this validation, 70% data of the dataset will go for training and 30% data will be for testing.

```
In [265]: xTrain, xTest, yTrain, yTest = train_test_split(vAR_Features_Train, obj_vAR_df["Final grade_cat"], test_size = 0.3,train_size=0.7

In [266]: xTrain.shape
Out[266]: (1400, 15)

In [267]: xTest.shape
Out[267]: (600, 15)

In [268]: vAR_model.fit(xTrain,yTrain)
Out[268]: LinearRegression()

In [269]: y_pred=vAR_model.predict(xTrain)
          y_pred
Out[269]: array([2.21970196, 2.25892252, 4.07516347, ..., 0.48646848, 2.21970196,
                 4.6428039 ])

In [270]: mean_squared_error(yTrain,y_pred)
Out[270]: 0.3300509021325271

In [271]: accuracy=vAR_model.score(xTrain,yTrain)
          accuracy=accuracy*100
          accuracy
Out[271]: 87.85957245224051
```

Probability

```
In [340]: from sklearn.model_selection import train_test_split
          X = vAR_df[['Assignment 1(%)','Assignment 2(%)','Assignment 3(%)','Semester 1(%)','Semester 2(%)',
                      'Backlogs','Tech-fest participation','Attendance','Quiz Voluntering',
                      'Maths Olympiad','Science olympiad','Adaptiveness','Secondary Language(Reading&writing)_cat',
                       'Social activity','Primary Language(Reading &writing)_cat']]
          y = vAR_df['Final grade']

          xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size=0.33, random_state=42)


          # fit Logistic Regression to the training set
          from sklearn.linear_model import LogisticRegression
          classifier = LogisticRegression(random_state = 0)
          classifier.fit(xTrain, yTrain)


          # predict the Test set results
          y_pred = classifier.predict(xTest)

          # make the confusion matrix for evaluation
          from sklearn.metrics import confusion_matrix
          confusion_matrix(yTest, y_pred)


          classifier.predict_proba(xTest)

          C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: 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[340]: array([[2.31498937e-01, 1.16511765e-01, 1.15288788e-01, ...,
                  4.68050959e-20, 1.28207786e-19, 2.25602696e-21],
                 [9.86183969e-01, 1.36889700e-02, 1.26899321e-04, ...,
                  1.46670325e-40, 2.60616031e-18, 4.29322792e-17],
                 [9.84589531e-01, 1.52397460e-02, 1.70554020e-04, ...,
                  1.67079373e-40, 3.12653443e-18, 4.91840696e-17],
                 ...,
                 [5.19097029e-05, 9.99948088e-01, 1.80662000e-09, ...,
                  6.02688776e-51, 1.36606008e-36, 9.42039957e-29],
                 [1.38147877e-02, 9.69814483e-01, 1.45877144e-02, ...,
                  3.35173321e-26, 2.84152172e-27, 1.15446039e-25],
                 [1.10890866e-01, 5.63505950e-03, 5.60805909e-01, ...,
                  1.46238054e-23, 8.78033118e-13, 2.51382644e-18]])
```

Confidence

```
In [ ]: import scipy.stats as st

In [342]: st.norm.interval(alpha=0.95, loc=np.mean(obj_vAR_df["Final grade_cat"]), scale=st.sem(obj_vAR_df["Final grade_cat"]))

Out[342]: (1.8539264412809229, 2.000073558719077)
```

80-20 Validation

In this validation, 80% data of the dataset will go for training and 20% data will be for testing.

```
In [251]: from sklearn.model_selection import train_test_split
          xTrain, xTest, yTrain, yTest = train_test_split(vAR_Features_Train, obj_vAR_df["Final grade_cat"], test_size = 0.2,train_size=0.8

In [252]: xTrain.shape
Out[252]: (1600, 15)

In [253]: xTest.shape
Out[253]: (400, 15)

In [254]: yTrain.shape
Out[254]: (1600,)

In [255]: yTest.shape
Out[255]: (400,)

In [256]: #reviewing learning algorithm
          y_pred=vAR_model.predict(xTrain)
          y_pred
Out[256]: array([2.18038017, 0.62707042, 0.45847043, ..., 0.13020746, 4.16097751,
                 0.46186586])

In [257]: from sklearn.metrics import mean_squared_error
          mean_squared_error(yTrain,y_pred)
Out[257]: 0.33598443512506043

In [258]: accuracy=vAR_model.score(xTrain,yTrain)
          accuracy=accuracy*100
          accuracy
Out[258]: 87.75943311435995
```

DeepSphere.AI
Enterprise AI and IIoT for Analytics

Probability

```
In [278]: from sklearn.model_selection import train_test_split
X = vAR_df[['Assignment 1(%)','Assignment 2(%)','Assignment 3(%)','Semester 1(%)','Semester 2(%)',
            'Backlogs','Tech-fest participation','Attendance','Quiz Volunteering',
            'Maths Olympiad','Science olympiad','Adaptiveness','Secondary Language(Reading&writing)_cat',
             'Social activity','Primary Language(Reading &writing)_cat']]
y = vAR_df['Final grade']

xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size=0.33, random_state=42)


# fit Logistic Regression to the training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(xTrain, yTrain)


# predict the Test set results
y_pred = classifier.predict(xTest)

# make the confusion matrix for evaluation
from sklearn.metrics import confusion_matrix
confusion_matrix(yTest, y_pred)


classifier.predict_proba(xTest)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: 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[278]: array([[2.31498937e-01, 1.16511765e-01, 1.15288788e-01, ...,
                  4.68050959e-20, 1.28207786e-19, 2.25602696e-21],
                 [9.86183969e-01, 1.36889700e-02, 1.26899321e-04, ...,
                  1.46670325e-40, 2.60616031e-18, 4.29322792e-17],
                 [9.84589531e-01, 1.52397460e-02, 1.70554020e-04, ...,
                  1.67079373e-40, 3.12653443e-18, 4.91840696e-17],
                 ...,
                 [5.19097029e-05, 9.99948088e-01, 1.80662000e-09, ...,
                  6.02688776e-51, 1.36606008e-36, 9.42039957e-29],
                 [1.38147877e-02, 9.69814483e-01, 1.45877144e-02, ...,
                  3.35173321e-26, 2.84152172e-27, 1.15446039e-25],
                 [1.10890866e-01, 5.63505950e-03, 5.60805909e-01, ...,
                  1.46238054e-23, 8.78033118e-13, 2.51382644e-18]])
```

Confidence

```
In [333]: import scipy.stats as st

In [335]: st.norm.interval(alpha=0.95, loc=np.mean(obj_vAR_df["Final grade_cat"]), scale=st.sem(obj_vAR_df["Final grade_cat"]))

Out[335]: (1.8539264412809229, 2.000073558719077)
```

DeepSphere.AI
Enterprise AI and IIoT for Analytics

90-10 Validation

In this validation, 90% data of the dataset will go for training and 10% data will be for testing.

```
In [354]: xTrain, xTest, yTrain, yTest = train_test_split(vAR_Features_Train, obj_vAR_df["Final grade_cat"], test_size = 0.1,train_size=0.

In [355]: xTrain.shape

Out[355]: (1800, 15)

In [356]: xTest.shape

Out[356]: (200, 15)

In [357]: vAR_model.fit(xTrain,yTrain)

Out[357]: LinearRegression()

In [358]: y_pred=vAR_model.predict(xTrain)
          y_pred

Out[358]: array([ 4.08862389,  0.45180984,  0.62460576, ...,  4.08862389,
                  2.33890187, -0.24459419])

In [359]: mean_squared_error(yTrain,y_pred)

Out[359]: 0.33300322735506105

In [360]: accuracy=vAR_model.score(xTrain,yTrain)
          accuracy=accuracy*100
          accuracy

Out[360]: 87.90056905048223
```

Probability

```
In [343]: from sklearn.model_selection import train_test_split
X = vAR_df[['Assignment 1(%)','Assignment 2(%)','Assignment 3(%)','Semester 1(%)','Semester 2(%)',
            'Backlogs','Tech-fest participation','Attendance','Quiz Volunteering',
            'Maths Olympiad','Science olympiad','Adaptiveness','Secondary Language(Reading&writing)_cat',
             'Social activity','Primary Language(Reading &writing)_cat']]
y = vAR_df['Final grade']

xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size=0.33, random_state=42)


# fit Logistic Regression to the training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(xTrain, yTrain)


# predict the Test set results
y_pred = classifier.predict(xTest)

# make the confusion matrix for evaluation
from sklearn.metrics import confusion_matrix
confusion_matrix(yTest, y_pred)


classifier.predict_proba(xTest)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: 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[343]: array([[2.31498937e-01, 1.16511765e-01, 1.15288788e-01, ...,
                  4.68050959e-20, 1.28207786e-19, 2.25602696e-21],
                 [9.86183969e-01, 1.36889700e-02, 1.26899321e-04, ...,
                  1.46670325e-40, 2.60616031e-18, 4.29322792e-17],
                 [9.84589531e-01, 1.52397460e-02, 1.70554020e-04, ...,
                  1.67079373e-40, 3.12653443e-18, 4.91840696e-17],
                 ...,
                 [5.19097029e-05, 9.99948088e-01, 1.80662000e-09, ...,
                  6.02688776e-51, 1.36606088e-36, 9.42039957e-29],
                 [1.38147877e-02, 9.69814483e-01, 1.45877144e-02, ...,
                  3.35173321e-26, 2.84152172e-27, 1.15446039e-25],
                 [1.10890866e-01, 5.63505950e-03, 5.60805909e-01, ...,
                  1.46238054e-23, 8.78033118e-13, 2.51382644e-18]])
```

Confidence

```
In [344]: import scipy.stats as st

In [345]: st.norm.interval(alpha=0.95, loc=np.mean(obj_vAR_df["Final grade_cat"]), scale=st.sem(obj_vAR_df["Final grade_cat"]))

Out[345]: (1.8539264412809229, 2.000073558719077)
```

Accuracy Visualisation

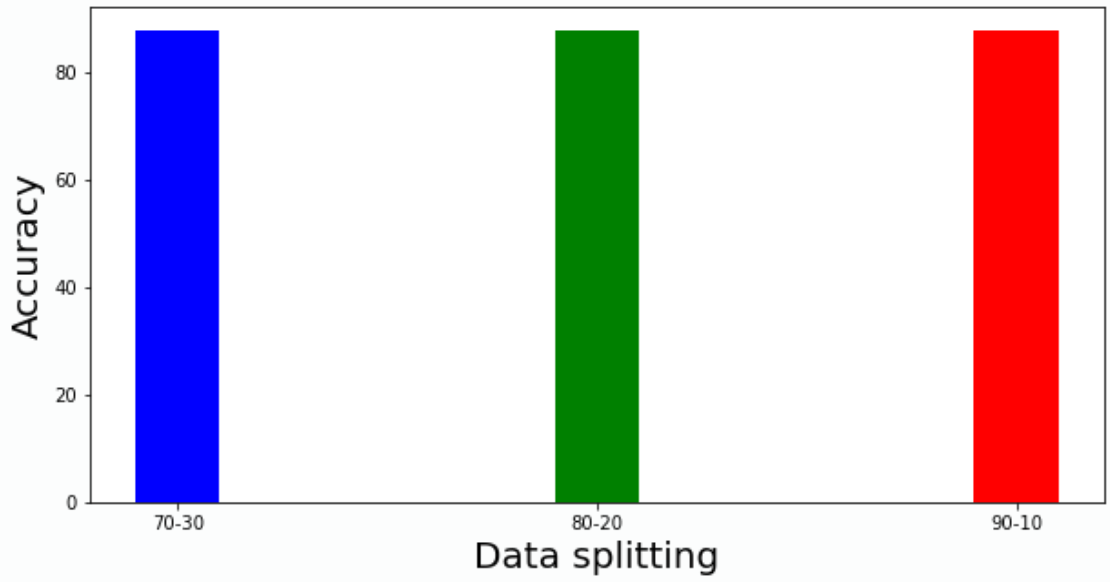
```
5]: Validation = ['70-30','80-20','90-10']
    Accuracy = [87.85,87.75,87.90]

    fig = vAR_plt.figure(figsize = (10,5))

    # creating the bar plot
    vAR_plt.bar(Validation,Accuracy, color =['blue','green','red'],width = 0.2)

    vAR_plt.xlabel("Data splitting",size=20)
    vAR_plt.ylabel("Accuracy",size=20)

    vAR_plt.show()
```



Weightage

Feature weighting is a technique which is used to estimating the importance of individual features .

```
In [289]: distribution = obj_vAR_df["Final grade_cat"]
          weights = vAR_df['Assignment 1(%)']

In [290]: def weighted_average_m1(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)
          weighted_average_m1(distribution,weights)
Out[290]: 1.69

In [292]: distribution = obj_vAR_df["Final grade_cat"]
          weights = vAR_df['Assignment 2(%)']

In [293]: def weighted_average_m1(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)
          weighted_average_m1(distribution,weights)
Out[293]: 1.65

In [295]: distribution = obj_vAR_df["Final grade_cat"]
          weights = vAR_df['Assignment 3(%)']

In [296]: def weighted_average_m1(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)
          weighted_average_m1(distribution,weights)
Out[296]: 1.6

In [297]: distribution = obj_vAR_df["Final grade_cat"]
          weights = vAR_df['Semester 1(%)']

In [298]: def weighted_average_m1(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)
          weighted_average_m1(distribution,weights)
Out[298]: 1.67

In [299]: distribution = obj_vAR_df["Final grade_cat"]
          weights = vAR_df['Semester 2(%)']

In [300]: def weighted_average_m1(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)
          weighted_average_m1(distribution,weights)
Out[300]: 1.62

In [303]: distribution = obj_vAR_df["Final grade_cat"]
          weights = vAR_df['Backlogs']

In [304]: def weighted_average_m1(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)
          weighted_average_m1(distribution,weights)
Out[304]: 4.29
```

```
In [305]: distribution = obj_WAR_df["Final grade_cat"]

In [306]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[306]: 1.25

In [307]: distribution = obj_WAR_df["Final grade_cat"]

In [308]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[308]: 1.81

In [311]: distribution = obj_WAR_df["Final grade_cat"]

In [312]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[312]: 1.11

In [313]: distribution = obj_WAR_df["Final grade_cat"]

In [314]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[314]: 0.15

In [315]: distribution = obj_WAR_df["Final grade_cat"]

In [316]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[316]: 0.5

In [317]: distribution = obj_WAR_df["Final grade_cat"]

In [318]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[318]: 0.98

In [319]: distribution = obj_WAR_df["Final grade_cat"]

In [320]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[320]: 0.19

In [321]: distribution = obj_WAR_df["Final grade_cat"]

In [322]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[322]: 1.52

In [324]: distribution = obj_WAR_df["Final grade_cat"]

In [325]: def weighted_average_nl(distribution,weights ):
              numerator=sum([distribution[i]*weights[i] for i in range (len(distribution))])
              denominator=sum(weights)
              return round(numerator/denominator,2)

Out[325]: 1.22
```

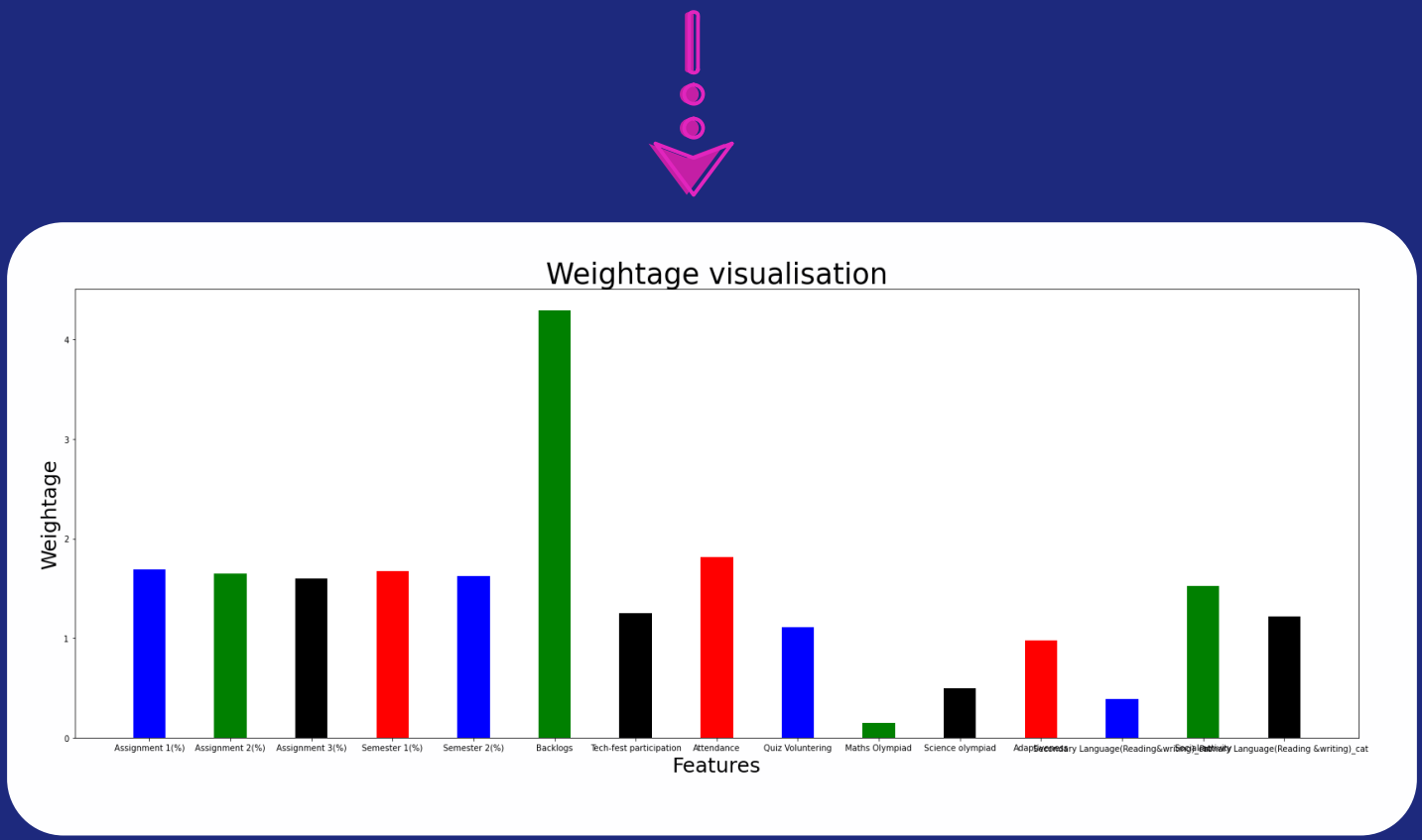
Weightage visualisation

```
In [54]: Features=['Assignment 1(%)','Assignment 2(%)','Assignment 3(%)','Semester 1(%)','Semester 2(%)',
                   'Backlogs','Tech-fest participation','Attendance','Quiz Voluntering',
                   'Maths Olympiad','Science olympiad','Adaptiveness','Secondary Language(Reading&writing)_cat',
                    'Social activity','Primary Language(Reading &writing)_cat']

         Weightage= [1.69,1.65,1.6,1.67,1.62,4.29,1.25,1.81,1.11,0.15,0.5,0.98,0.39,1.52,1.22]

         fig = vAR_plt.figure(figsize = (28,10))

         # creating the bar plot
         vAR_plt.bar(Features, Weightage, color =['cyan','red','blue','green','cyan','black','red','blue','green','black'],width = 0.4)
         vAR_plt.xlabel("Features",size=25)
         vAR_plt.ylabel("Weightage",size=25)
         vAR_plt.title("Weightage visualisation",size=35)
         vAR_plt.show()
```



Who We Are

The DeepSphere.AI team comprises MIT learning facilitators, University of California instructors, Harvard PhDs, Stanford alumni, industry leaders, and proven entrepreneurs. The group collectively brings business and technology together for in-depth, hands-on AI learning and a risk-free implementation and AI adoption.

Our Team

Massachusetts Institute of Technology

Harvard

Stanford University

The University of California

"Jothi...I am honored to learn from your comments and messages in the MIT Sloan&CSAIL course. I am deeply and impressed inspired by your ideas which make a great impact on my learning path... "
SinTing (Adele) Lui
MIT CSAIL AI Student

DeepSphere.AI
Enterprise AI and IIoT for Analytics

Next Steps

Contact

DeepSphere.AI, Inc.
2100 Geng Road, Suite 210
Palo Alto, CA 94303
USA
Info@ DeepSphere.AI

DeepSphere.AI is the first and the ONLY subscription-based comprehensive applied artificial intelligence platform for learning artificial intelligence at the deepest level

DeepSphere.AI
Enterprise AI and IIoT for Analytics