DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-324
Course Title: Machine Learning
Complex Engineering Problem
TE Batch 2019, Spring Semester 2022
Grading Rubric
TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1	**Zobia Khan**	**CS-19063**
S2	**Muqadas Ashraf**	**CS-19053**
S3	-	-

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-2, CPA-3) **[8 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 2: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 3: Does the report adhere to the given format and requirements? **[6 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
Criterion 4: How does the student performed individually and as a team member? (CPA-1, CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			

Final Score = (Criterial_1_score x 2) + (Criteria_2_score / 2) + (Criteria_3_score x (3/2)) + (Criteria_4_score)
= _____

Introduction:

Cumulative Grade Point Average (CGPA) refers to the overall Grade Point Average (GPA), obtained by dividing the total Grade Points (GPs) earned in all courses attempted by the total degree-credit hours in all attempted courses. Cumulative Grade Point Average (CGPA) prediction is an important area for understanding education performance trend of students and identifying the demographic attributes to devise effective educational strategies and infrastructure. This project aims to analyze the accuracy of CGPA prediction of students resulted from predictive models, We have used three models, Model 1 predicts final CGPA based on GPs of first year only, Model 2 predicts final CGPA based on GPs of first two years, Model 3 predict final CGPA based on GPs of first three years.

Data Description:

The data set used in this project is that of university students contains the grades of students in different courses studied in 4 years in csv format. Dataset contains grades of 571 students in 42 courses.

Data Preprocessing:

- In start we read our data by **pd.read_csv** and using **dataset.info()** we get the information of our data which shows that we have some null values in our data.
- Now for training purpose we split our input and output features and drop those features which don't contribute in output, for this we use **dataset.drop(['CGPA','Seat No.'],axis="columns")** command as CGPA is output and Seat No is irrelevant column by this we get our input features.
- As we have Grades in our dataset which is string so we convert these grades into float values by assigning GPs.

```
In [6]: AssignGPs={
            "A+": 4.0,
            "A": 4.0,
            "A-" : 3.7,
            "B+" : 3.4,
            "B" : 3.0,
            "B-" : 2.7,
            "C+" : 2.4,
            "C" : 2.0,
            "C-" : 1.7,
            "D+" : 1.4,
            "D" : 1.0,
            "F" : 0.0,
            "WU" :0
        }
```

- Now we are checking for null values using **courses.columns[courses.isna().any()]** command, this will print all those features which has null values and we fill 0 to null values using function **courses[0:].fillna(0)**
- Grade points are then multiplied with the credit hours of course.

Models:

We made 3 models, First model focuses on 1st years courses, Second Model focuses on 1st and 2nd year courses and the Third Model focuses on 1st,2nd and 3rd year courses. In each model we implement 2 algorithms Linear Regression and Support Vector Regression.

In all models we implement two Machine Learning Algorithms, Linear Regression and then SVR (Support Vector Regressor.

We import linear regression using **from sklearn.linear_model import LinearRegression** command, also for training testing purpose we import train test split using **from sklearn.model_selection import train_test_split** command.The test size for all models is 0.2 that is 80% data is for training and the 20% for testing.

Finally, we use our Linear Regression model for prediction, we predict using **predictions_lr=model_lr.predict(X_test)** command.

```
model_lr=LinearRegression()
```

```
model_lr.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Now for second algorithm we import SVR using **from sklearn.svm import SVR** command. Finally, we use our SVR model for prediction, we predict using predictions_svr=model_svr.predict(X_test) command.

```
from sklearn.svm import SVR
model_svr=SVR()
```

```
model_svr.fit(X_train,y_train)
```

```
▾ SVR
SVR()
```

Model Evaluation:

The evaluation metrices used in all models to check the their performance are: RMSE(Root Mean Squared Error), sklearn bult-in function score() and cross validation score.

1) RMSE:

Root mean squared error (RMSE) is the square root of the mean of the square of all of the residuals(Actual – Predicted).Residuals are a measure of how far from the regression line data points are. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is most useful when large errors are particularly undesirable. It indicates the absolute fit of the model to the data.They are negatively-oriented scores, which means lower values are better.

$$RMSE = \sqrt{\frac{1}{N}\sum\left(\hat{y}_i - y_i\right)^2}$$

2)SCORE:

Sci-kit learn in python model. score automates the prediction of your data using X_test and compares it with Y_test and by default uses the R-squared metric to so we don't need to manually derive prediction from the model. It indicates how better the algorithm is for our model, higher the value of score better the model is.

3)CROSS VALIDATION SCORE:

Cross-Validation is basically a resampling technique to make our model sure about its efficiency and accuracy on the unseen data. In our models, cross validation is applied on test data with cv = 10, Which means data is split 10 times and its score is calculated in each iteration, the function returns the list of scores calculated in each iteration. This evaluation metric will help in comparing models.

Model 1:

This model contain grades 11 courses, number of entries are 571,

```
In [ ]: first_year=courses.loc[:, "PH-121":"CS-107"]

In [ ]: first_year.head()
```

Out[15]:

	PH-121	HS-101	CY-105	HS-105/12	MT-111	CS-105	CS-106	EL-102	EE-119	ME-107	CS-107
0	10.8	2.8	5.1	6.0	5.1	5.6	4.0	5.1	10.8	6.8	6.8
1	16.0	2.0	4.2	3.0	8.1	8.0	4.0	12.0	5.6	4.0	13.6
2	16.0	6.0	12.0	8.1	10.2	16.0	10.8	10.2	14.8	14.8	10.8
3	4.0	4.8	4.2	3.0	3.0	14.8	5.6	5.1	4.0	9.6	4.0
4	14.8	7.4	11.1	10.2	12.0	16.0	14.8	10.2	16.0	14.8	8.0

Performance of model 1:

Linear Regression:

```
In [ ]: from sklearn.metrics import mean_squared_error
        mse=mean_squared_error(y_test,predictions_lr)
        rmse=np.sqrt(mse)
        rmse
Out[23]: 0.2935811341668194

In [ ]: from sklearn.model_selection import cross_val_score
        cross_value=cross_val_score(model_lr, X_test,y_test,cv=10)
        results.append(cross_value)
        cross_value
Out[24]: array([0.82067555, 0.73280906, 0.81602184, 0.8359261 , 0.7654803 ,
               0.62058804, 0.5696741 , 0.73790031, 0.77699119, 0.58003835])

In [ ]: model_lr.score(X_test,y_test)
Out[25]: 0.7852037261945182
```

Support Vector Regressor:

```
In [ ]: from sklearn.metrics import mean_squared_error
        mse=mean_squared_error(y_test,predictions_svr)
        rmse=np.sqrt(mse)
        rmse
Out[37]: 0.2565595256838977

In [ ]: model_svr.score(X_test,y_test)
Out[38]: 0.8359611396730238

In [ ]: from sklearn.model_selection import cross_val_score
        cross_value=cross_val_score(model_svr, X_test,y_test,cv=10)
        results.append(cross_value)
        cross_value
Out[39]: array([0.8426795 , 0.86021066, 0.80494057, 0.84547024, 0.7706144 ,
               0.69129797, 0.56530449, 0.74162255, 0.75036547, 0.79631834])
```

Model 2:

This model contain grades 22 courses, number of entries are 571

```
In [ ]: second_year.head()
```

Out[41]:

	PH-121	HS-101	CY-105	HS-105/12	MT-111	CS-105	CS-106	EL-102	EE-119	ME-107	...	MT-222	EE-222	MT-224	CS-210	CS-211	CS-203	CS-214	EE-2
0	10.8	2.8	5.1	6.0	5.1	5.6	4.0	5.1	10.8	6.8	...	3.0	14.8	8.1	7.2	4.2	1.4	4.0	16
1	16.0	2.0	4.2	3.0	8.1	8.0	4.0	12.0	5.6	4.0	...	8.1	6.8	4.2	4.2	3.0	2.0	4.0	10
2	16.0	6.0	12.0	8.1	10.2	16.0	10.8	10.2	14.8	14.8	...	12.0	16.0	12.0	12.0	12.0	4.0	14.8	16
3	4.0	4.8	4.2	3.0	3.0	14.8	5.6	5.1	4.0	9.6	...	4.2	10.8	3.0	6.0	10.2	2.4	4.0	5
4	14.8	7.4	11.1	10.2	12.0	16.0	14.8	10.2	16.0	14.8	...	12.0	14.8	11.1	11.1	8.1	4.0	14.8	16

5 rows × 22 columns

Performance of model 2:

Linear Regression:

```
In [ ]: from sklearn.metrics import mean_squared_error
        mse=mean_squared_error(y_test,predictions_lr2)
        rmse=np.sqrt(mse)
        rmse
```

```
Out[48]: 0.20074477029324156
```

```
In [ ]: from sklearn.model_selection import cross_val_score
        cross_value=cross_val_score(model_lr2, X_test,y_test,cv=10)
        results.append(cross_value)
        cross_value
```

```
Out[49]: array([ 0.91274888,  0.8965322 ,  0.89221447,  0.87998068,  0.71703154,
                 0.90453592,  0.89016718,  0.89096178,  0.91590503, -0.18450605])
```

```
In [ ]: model_lr2.score(X_test,y_test)
```

```
Out[50]: 0.8801032165244438
```

Support Vector Regressor:

```
In [ ]: from sklearn.metrics import mean_squared_error
        mse=mean_squared_error(y_test,predictions_svr2)
        rmse=np.sqrt(mse)
        rmse
```

```
Out[62]: 0.16389820616069348
```

```
In [ ]: model_svr2.score(X_test,y_test)
```

```
Out[63]: 0.920077795834987
```

```
In [ ]: from sklearn.model_selection import cross_val_score
        cross_value=cross_val_score(model_svr2, X_test,y_test,cv=10)
        results.append(cross_value)
        cross_value
```

```
Out[64]: array([0.90573502, 0.90277648, 0.92677237, 0.92761915, 0.84919055,
                0.85600804, 0.91769396, 0.95723852, 0.89698997, 0.8803421 ])
```

Model 3:

This model contain grades 33 courses, number of entries are 571

```
In [43]: Third_year.head()
```

Out[43]:

	PH-121	HS-101	CY-105	HS-105/12	MT-111	CS-105	CS-106	EL-102	EE-119	ME-107	...	EF-303	HS-304	CS-301	CS-302
0	10.8	2.8	5.1	6.0	5.1	5.6	4.0	5.1	10.8	6.8	...	5.1	4.8	12.0	6.0
1	16.0	2.0	4.2	3.0	8.1	8.0	4.0	12.0	5.6	4.0	...	3.0	3.4	14.8	4.2
2	16.0	6.0	12.0	8.1	10.2	16.0	10.8	10.2	14.8	14.8	...	7.2	8.0	13.6	11.1
3	4.0	4.8	4.2	3.0	3.0	14.8	5.6	5.1	4.0	9.6	...	3.0	5.4	4.0	8.1
4	14.8	7.4	11.1	10.2	12.0	16.0	14.8	10.2	16.0	14.8	...	5.1	7.4	10.8	11.1

5 rows × 33 columns

Performance of model 3:

Linear Regression:

```
In [48]: # Calculating RMSE
         mse=mean_squared_error(y_test,predictions_lr3)
         rmse=np.sqrt(mse)
         rmse

Out[48]: 0.13801247043510861

In [49]: # Test data is cross validated with CV value 10 which returns the array of scores
         cross_value=cross_val_score(model_lr3, X_test,y_test,cv=10)
         results.append(cross_value)
         cross_value

Out[49]: array([0.9265165 , 0.97249496, 0.94997935, 0.97198021, 0.87630457,
                0.97768429, 0.98255224, 0.94969403, 0.98157847, 0.98280352])

In [50]: # Calculating accuracy of the model
         model_lr3.score(X_test,y_test)

Out[50]: 0.9542519541675104
```

Support Vector Regressor:

```
In [54]: # Calculating RMSE
         mse=mean_squared_error(y_test,predictions_svr3)
         rmse=np.sqrt(mse)
         rmse

Out[54]: 0.12024272894728873

In [55]: # Calculating accuracy of the model
         model_svr3.score(X_test,y_test)

Out[55]: 0.9652740979342452

In [56]: # Test data is cross validated with CV value 10 which returns the array of scores
         cross_value=cross_val_score(model_svr3, X_test,y_test,cv=10)
         results.append(cross_value)
         cross_value

Out[56]: array([0.95169267, 0.97532471, 0.94150481, 0.88471334, 0.9197064 ,
                0.98502732, 0.94286854, 0.9607391 , 0.96318118, 0.9496501 ])
```

Overall Review:

Best predictions are given by model 3 because it has more data (features) than any model which give best training and testing accuracy and in this model best algorithm is Support Vector Regression.

Algorithms:

Linear Regression

Linear regression is a Linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x), also this is used for regression problems. Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available.

Hypothesis function for Linear Regression is:

$Y = \theta_0 + \theta_1 x$

Where θ_0 is the intercept and θ_1 is coefficient of independent variable x.

While training the model we are given :
x: input training data
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.

θ_1: intercept

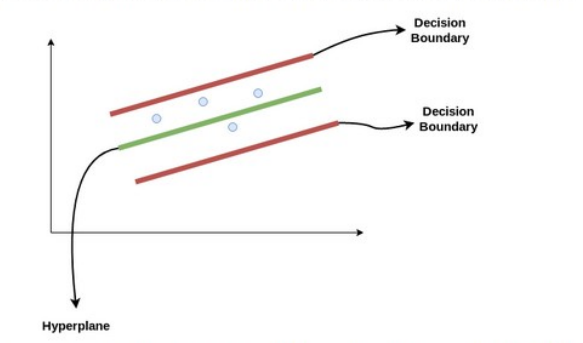
θ_2: coefficient of x

Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x

To update θ_1 and θ_2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively updating the values, reaching minimum cost.

Support Vector Regression

Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems, this algorithm supports both linear and non-linear regressions. The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample. SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data. In simple regression, the idea is to minimize the error rate while in SVR the idea is to fit the error inside a certain threshold which means, work of SVR is to approximate the best value within a given margin called decision boundry Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line**.** Our best fit line is the hyperplane that has a maximum number of points.



The decision boundary lines are at any distance, say ‘a’, from the hyperplane. So, these are the lines that we draw at distance ‘+a’ and ‘-a’ from the hyperplane. This ‘a’ in the text is basically referred to as epsilon.

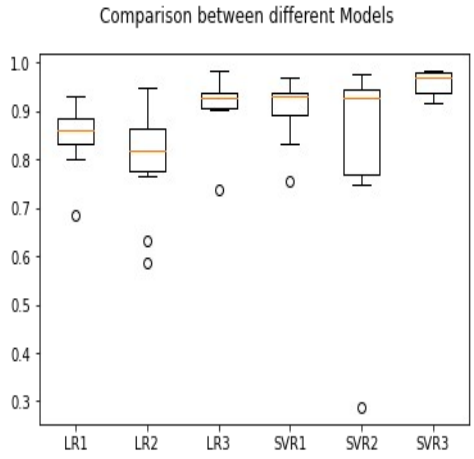
Assuming that the equation of the hyperplane is as follows: $Y = wx+b$ (equation of hyperplane)

Then the equations of decision boundary become: $wx+b= +a$, $wx+b= -a$

Thus, any hyperplane that satisfies our SVR should satisfy: $-a < Y- wx+b < +a$

Our main aim here is to decide a decision boundary at ‘a’ distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line. Hence, we are going to take only those points that are within the decision boundary and have the least error rate, or are within the Margin of Tolerance. This gives us a better fitting model.

Graphical Comparison:



Our models are compared using ‘Box and Whisker’ plot. From left, first 3 models are implemented on First year,Second year and Third year courses using ‘Linear Regression’ algorithm (LR1,LR2,LR3) and last 3 models are implemented using ‘Support Vector Regression’ algorithm (SVR1,SVR2,SVR3).Theses models are compared based on their cross validated scores. The scores close to 1.0 are considered as good models and the graph shows that performance of all models are good, scores of our models lies between 0.7 to 1.0, there are some low score values plotted as dots on graph but very few. Here the best score is of model ‘SVR3’ because its cross validated scores lies very close to score 1.0 than other models.

Addressing Over-fitting and Under-fitting Issues:

Overall the performance of all our model is good because based on the evaluations of our model the training and testing scores are high, which indicates that our models have succeeded in generalization which means that any unseen data samples produces correct predictions. All the models have no issue of underfitting or overfitting. We have also applied some steps to overcome over-fitting, under-fitting issues.

Under-fitting:

If our model is under-fit then the model has fit on data poorly and most of the predictions are wrong, training and testing accuracy are very low.In order to overcome these problem we start adding more features in our data, first we check scores for First model using First year courses as features and then we increase no of courses and develop Second model and Third model. We train every model and their scores turned out to be high for the model with large number of features.

Below is the tabular comparison of all models using Linear Regression.

MODELS	TEST SCORE
MODEL 1	0.842
MODEL 2	0.845
MODEL 3	0.952

Below is the tabular comparison of all models using Support Vector Regression.

MODELS	TEST SCORE
MODEL 1	0.851
MODEL 2	0.938
MODEL 3	0.934

Over-fitting:

Over-fitting problem occurs if our model has failed to generalize the model, predictions on unseen data is wrong, which means that training score is high but testing score is very low. So to avoid this issue we used cross-validation in each algorithm for every model in which we train our model in 10 folds so in each fold during training our model have some new data set so now it no more completely rely on fixed train dataset and some old one and in each fold we calculate score so in end we get array of scores and these values are more accurate than individual scores of model because here we train our model on whole data set so the difference between its predicted output and expected output is less because by cross-validation we made our model more general.

Linear Regression:

MODELS	CROSS VALIDATION SCORE
MODEL 1	[0.72014214, 0.81264185, 0.85894866, 0.74687861, 0.86982078,0.86735222, 0.75549386, 0.79470879, 0.70548586, 0.76274108]
MODEL 2	[0.86767026, 0.857238 , 0.84883619, 0.80342767, 0.90421597,0.89379617, 0.97824198, 0.87714191, 0.96555115, 0.91096506]
MODEL 3	[0.97426557, 0.86855101, 0.45598116, 0.91948966, 0.54121032, 0.73554201, 0.86526851, 0.86082839, 0.9245787 , 0.40974013]

Support Vector Regression:

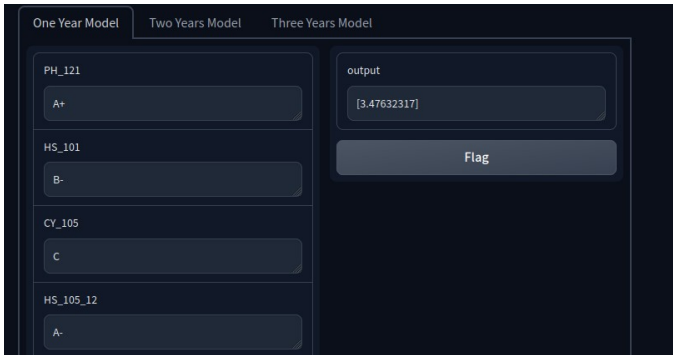
MODELS	CROSS VALIDATION SCORE
MODEL 1	[0.86185166, 0.89136512, 0.88961475, 0.79966914, 0.65474681,0.8890789 , 0.75682992, 0.81217861, 0.5339139 , 0.75535639]
MODEL 2	[0.92484957, 0.92547596, 0.80398006, 0.90260556, 0.9233244 ,0.90711561, 0.94039602, 0.88447276, 0.93806112, 0.91402246]
MODEL 3	[0.98690064, 0.95078648, 0.87200147, 0.97773952, 0.93867517,0.96420708, 0.98226575, 0.97484013, 0.95198317, 0.95674866]

Distinguishing Feature:

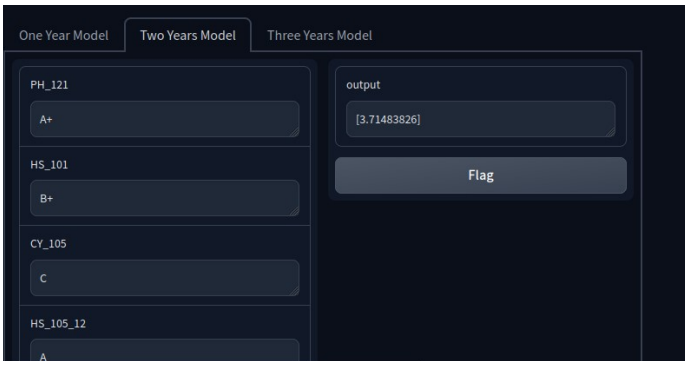
We have developed a user interface for predicting the CGPA of student using ‘Gradio’ library. User can select First year, Second year or Third year and input the grades on respective courses, output is generated in the form of CGPA of student.

We used Support Vector Regression model to make prediction because its training and testing scores are better than Linear Regression.

MODEL 1:



MODEL 2:



MODEL 3: