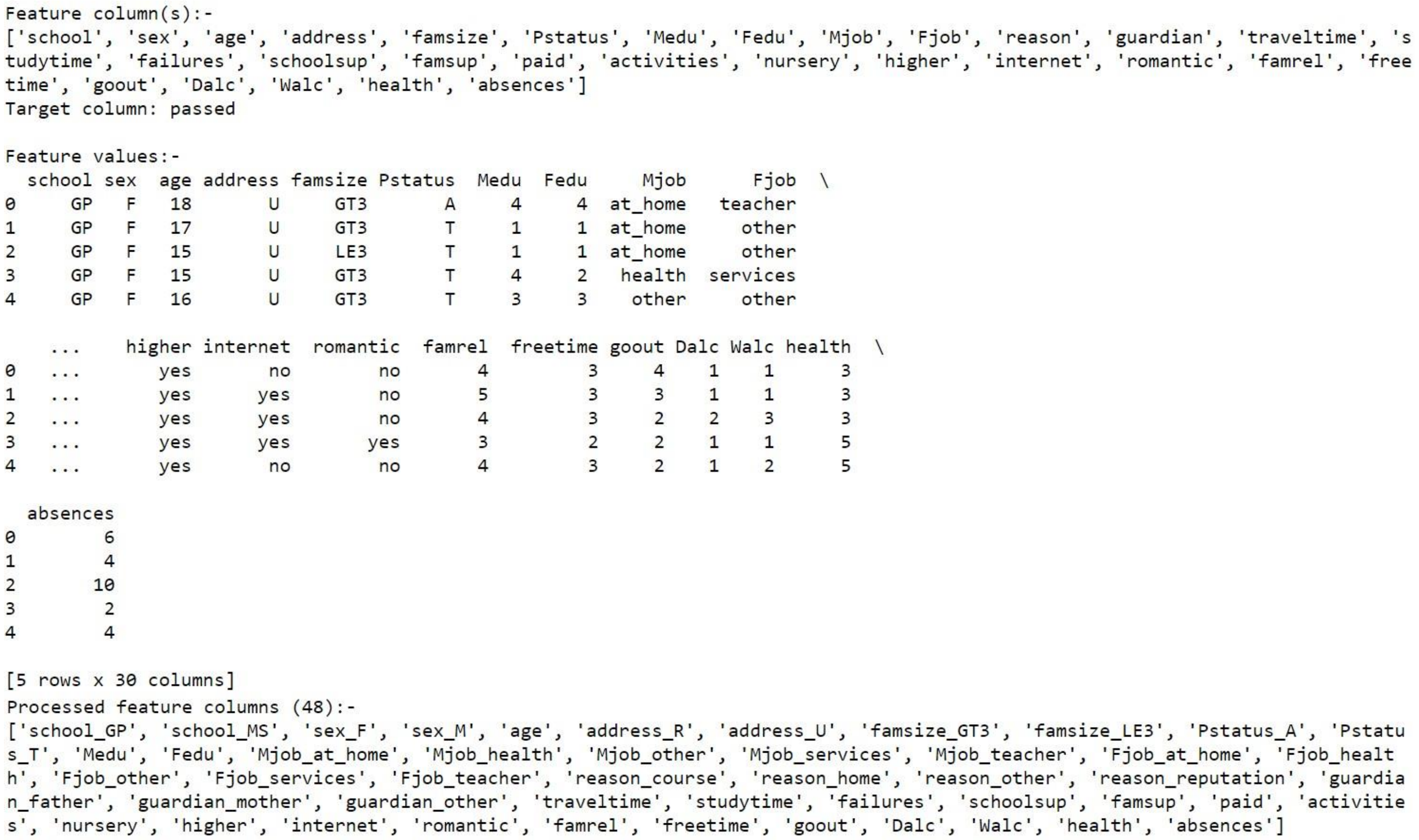
Classification vs Regression.

Students who might need early intervention is a classification problem because the output that we will get is Boolean expression whether student need or does not need early intervention.

Exploring the Data.

Total number of students: 395
Number of students who passed: 265
Number of students who failed: 130
Number of features: 30
Graduation rate of the class: 67.09%

Preparing the Data.



```
Feature column(s):-
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 's
tudytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'free
time', 'goout', 'Dalc', 'Walc', 'health', 'absences']
Target column: passed

Feature values:-
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher
1     GP   F   17       U     GT3       T     1     1  at_home     other
2     GP   F   15       U     LE3       T     1     1  at_home     other
3     GP   F   15       U     GT3       T     4     2   health  services
4     GP   F   16       U     GT3       T     3     3    other     other

    ...    higher internet  romantic  famrel  freetime goout Dalc Walc health  \
0   ...       yes       no        no       4         3     4    1    1      3
1   ...       yes      yes        no       5         3     3    1    1      3
2   ...       yes      yes        no       4         3     2    2    3      3
3   ...       yes      yes       yes       3         2     2    1    1      5
4   ...       yes       no        no       4         3     2    1    2      5

  absences
0        6
1        4
2       10
3        2
4        4

[5 rows x 30 columns]
Processed feature columns (48):-
['school_GP', 'school_MS', 'sex_F', 'sex_M', 'age', 'address_R', 'address_U', 'famsize_GT3', 'famsize_LE3', 'Pstatus_A', 'Pstatu
s_T', 'Medu', 'Fedu', 'Mjob_at_home', 'Mjob_health', 'Mjob_other', 'Mjob_services', 'Mjob_teacher', 'Fjob_at_home', 'Fjob_healt
h', 'Fjob_other', 'Fjob_services', 'Fjob_teacher', 'reason_course', 'reason_home', 'reason_other', 'reason_reputation', 'guardia
n_father', 'guardian_mother', 'guardian_other', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activitie
s', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']
```

Training and Evaluating Models.

I chose the following models:

1) K Neighbors Classifier
2) Logistic Regression
3) Support Vector Classification

Because each of them can solve Classification problem. Each model has its own advantages and disadvantages.

Test name	Advantages	Disadvantages
K Neighbors Classifier	Simplicity	Poor run time performance when training set is large
	Effectiveness	Very sensitive to irrelevant or redundant features
	Good classification performance	Computation cost is high
	Robust to noisy training data	
	Effective to large training data	
Logistic Regression	Simplicity	Requires huge amount of data to get stable, meaningful results.
	Low computation cost	
SVC	Effective in high dimensional spaces.	If number of features is greater than number of samples this method gives poor performance
	Effective in cases where number of dimensions is greater than the number of samples.	Do not directly provide probability estimates
	Memory efficient.	
	Versatile	

Test name	Training time (secs)	Prediction time (secs)	F1 score for training set	Prediction time (secs)	F1 score for test set	Training set size	Testing set size
KNeighborsClassifier	0.001	0.004	0.87012987	0.006	0.760820046	100	295
LogisticRegression	0.002	0.001	0.92	0.001	0.763888889		
SVC	0.002	0.001	0.893081761	0.003	0.786469345		

Test name	Training time (secs)	Prediction time (secs)	F1 score for training set	Prediction time (secs)	F1 score for test set	Training set size	Testing set size
KNeighborsClassifier	0.001	0.005	0.852348993	0.007	0.753521127	200	195
LogisticRegression	0.005	0.001	0.845360825	0.001	0.788321168		
SVC	0.004	0.003	0.888157895	0.003	0.78807947		

Test name	Training time (secs)	Prediction time (secs)	F1 score for training set	Prediction time (secs)	F1 score for test set	Training set size	Testing set size
KNeighborsClassifier	0.001	0.011	0.864988558	0.005	0.759124088	300	95
LogisticRegression	0.005	0.001	0.850574713	0.001	0.759124088		
SVC	0.008	0.006	0.883116883	0.002	0.775510204		

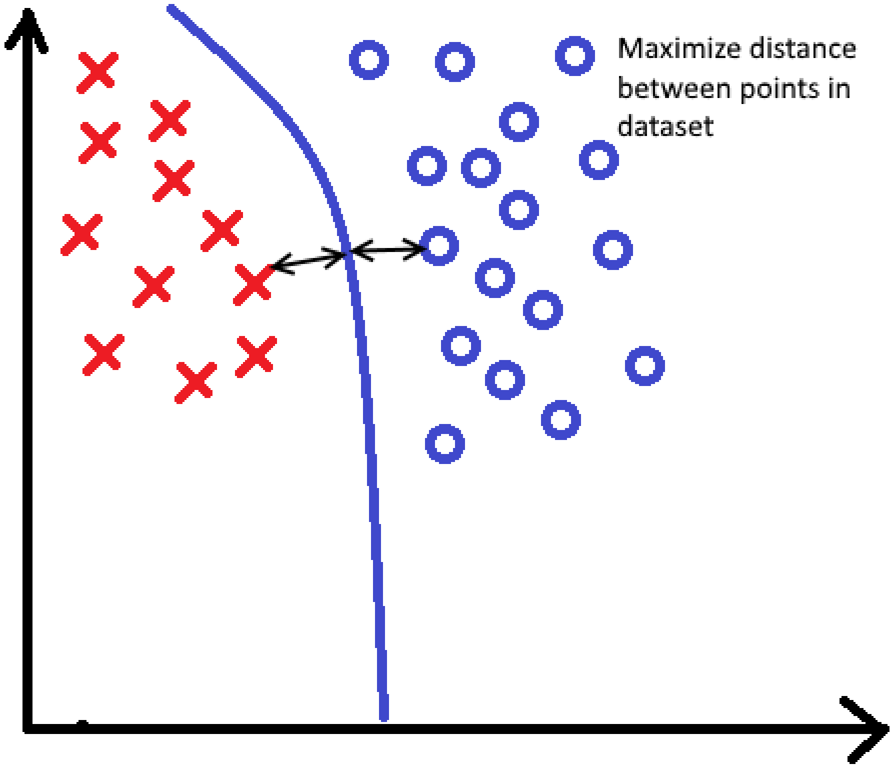
Average					
Test name	Training time (secs)	Prediction time (secs)	F1 score for training set	Prediction time (secs)	F1 score for test set
KNeighborsClassifier	0.001	0.006666667	0.862489141	0.006	0.757821753
LogisticRegression	0.004	0.001	0.871978512	0.001	0.770444715
SVC	0.004666667	0.003333333	0.888118846	0.002666667	0.783353006

Choosing the Best Model.

SVC model has the highest average score. However, it has the worst training time and average performance time. So if we have limited resources I do not think that it is the best model however in given problem I think that accuracy is more important than computational time. Therefore, I think that SVC model is the best model that I can choose for solving given problem. However, if we had limited resources and computational time mattered for us, I would choose Logistic Regression because it is more balanced.

```
The best model is SVC(C=1, cache_size=200, class_weight=None, coef0=0.0, degree=2, gamma=0.0,
  kernel='poly', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
0.903225806452
```

SVM takes the data and find the line that is maximum from all the data points in dataset.



Since the data can be complicated, we cannot draw a straight line to separate the data. Therefore, in some cases we need to add some new things (“features”, “dimensions”) that will help us to draw a straight line to separate the data instead of circle, square, islands etc. These functions, which helps us to do it, are named kernels.

```
The models final f1 scores for training set: 0.889908256881
The models final f1 scores for test set: 0.782608695652
```