

## Import Dataset

```
In [1]: import numpy as np
import pandas as pd
```

```
In [2]: data = pd.read_csv(r'data/student-mat.csv', sep = ';')
```

```
In [3]: data.head()
```

Out[3]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famre
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4
3	GP	F	15	U	GT3	T	4	2	health	services	...	3
4	GP	F	16	U	GT3	T	3	3	other	other	...	4

5 rows × 33 columns



```
In [4]: data.shape
```

Out[4]: (395, 33)

## Pre-processing

G1: First year grade

G2: Second year grade

G3: Third year grade (target variables)

: G3 is highly correlated with G1, G2

### Range of G3 from 0-20 to 0-4

```
In [5]: data['G3'] = data['G3'] // 5
```

In [6]: `data.head()`

Out[6]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famre
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4
3	GP	F	15	U	GT3	T	4	2	health	services	...	3
4	GP	F	16	U	GT3	T	3	3	other	other	...	4

5 rows × 33 columns



In [7]: `data = data.reindex(np.random.permutation(data.index))`

### one-hot encoding (Not numeric variables)

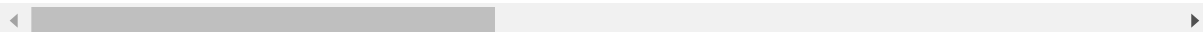
In [8]: `data = pd.get_dummies(data)`

In [9]: `data.head()`

Out[9]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	a
250	18	3	2	2	1	1	4	4	5	2	...	1
253	16	2	1	2	1	0	3	3	2	1	...	0
321	17	2	2	1	2	0	4	2	2	1	...	1
238	17	2	1	3	2	0	2	1	1	1	...	0
339	17	3	2	1	2	0	4	3	3	2	...	1

5 rows × 59 columns



In [10]: `X = data.drop(['G3'], axis = 1)`

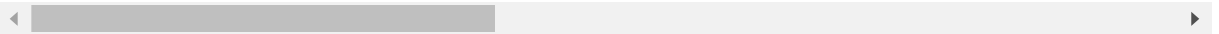
In [11]: `y = data['G3']`

In [12]: X.head()

Out[12]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	...
<b>250</b>	18	3	2	2	1	1	4	4	5	2	...	1
<b>253</b>	16	2	1	2	1	0	3	3	2	1	...	0
<b>321</b>	17	2	2	1	2	0	4	2	2	1	...	1
<b>238</b>	17	2	1	3	2	0	2	1	1	1	...	0
<b>339</b>	17	3	2	1	2	0	4	3	3	2	...	1

5 rows × 58 columns



In [13]: y.head()

Out[13]: 250 1  
 253 1  
 321 1  
 238 2  
 339 2  
 Name: G3, dtype: int64

## Classifiers

```
In [14]: from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
```

## Classifier List

1. OneR
2. Nearest Neighbors
3. Linear SVM (Kernel: Linear)
4. RBF SVM (Kernel: RBF)
5. Decision Tree
6. Random Forest
7. Neural Network
8. AdaBoost (Ensemble based on the Decision Tree classifier)
9. Gaussian Naive Bayes

```
In [15]: names = ["OneR", "Nearest Neighbors", "Linear SVM", "RBF SVM",
                  "Decision Tree", "Random Forest", "Neural Net", "AdaBoost", "Naive Bayes"]
```

```
In [16]: classifiers = [  
    DecisionTreeClassifier(max_depth=1),  
    KNeighborsClassifier(3),  
    SVC(kernel="linear", C=0.025),  
    SVC(kernel="rbf", C=0.025),  
    DecisionTreeClassifier(max_depth=10),  
    RandomForestClassifier(max_depth=10, n_estimators=300),  
    MLPClassifier(alpha=1, hidden_layer_sizes = (100, 50, 30)),  
    AdaBoostClassifier(n_estimators = 500),  
    GaussianNB()]
```

```
In [17]: from sklearn.model_selection import cross_validate  
from sklearn.model_selection import cross_val_predict  
from sklearn.metrics import precision_recall_fscore_support
```

```
In [18]: evaluations = []
```

## ZeroR

```
In [19]: max_index = data.groupby(['G3'])['G3'].count().idxmax()
```

```
In [20]: evaluations.append(['ZeroR', data.groupby(['G3'])['G3'].count()[max_index] / len(y)])
```

## Using Classifier List above

```
In [21]: for name, clf in zip(names, classifiers):  
         y_pred = cross_val_predict(clf, X, y, cv = 10)  
         precision_recall_f1 = precision_recall_fscore_support(y, y_pred, average='weighted')  
         scores = cross_validate(clf, X, y, cv = 10)  
         evaluations.append([name, scores['test_score'].mean(), precision_recall_f1[:3]])
```

6/11

7/11

```

reached and the optimization hasn't converged yet.
% self.max_iter, ConvergenceWarning)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/neural_network/multilayer_perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200)
reached and the optimization hasn't converged yet.
% self.max_iter, ConvergenceWarning)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/neural_network/multilayer_perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200)
reached and the optimization hasn't converged yet.
% self.max_iter, ConvergenceWarning)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/neural_network/multilayer_perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200)
reached and the optimization hasn't converged yet.
% self.max_iter, ConvergenceWarning)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/model_selection/_split.py:605: Warning: The least populated class in y has only 1 members, which is too few. The
minimum number of members in any class cannot be less than n_splits=10.
% (min_groups, self.n_splits)), Warning)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/model_selection/_split.py:605: Warning: The least populated class in y has only 1 members, which is too few. The
minimum number of members in any class cannot be less than n_splits=10.
% (min_groups, self.n_splits)), Warning)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/model_selection/_split.py:605: Warning: The least populated class in y has only 1 members, which is too few. The
minimum number of members in any class cannot be less than n_splits=10.
% (min_groups, self.n_splits)), Warning)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/metrics/classification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)
/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/model_selection/_split.py:605: Warning: The least populated class in y has only 1 members, which is too few. The
minimum number of members in any class cannot be less than n_splits=10.
% (min_groups, self.n_splits)), Warning)

```

```

In [22]: evaluations.sort(key = lambda evaluations: evaluations[1])
evaluations.reverse()

```



```
In [23]: for evaluation in evaluations:
          print("-----" * 5)
          print(evaluation[0])
          print("Accuracy: ", evaluation[1])
          if len(evaluation) > 2: print("Precision, Recall, F1", evaluation[2])
          print("-----" * 5, end = "WnWn")
```

---

Linear SVM

Accuracy: 0.8551918158763548

Precision, Recall, F1 (0.853963429550741, 0.8556962025316456, 0.8546268327816859)

---

AdaBoost

Accuracy: 0.8263845671598604

Precision, Recall, F1 (0.8620274562615824, 0.8278481012658228, 0.8243937647474007)

---

Neural Net

Accuracy: 0.8224507284617146

Precision, Recall, F1 (0.8185230426932188, 0.8202531645569621, 0.8193130659197408)

---

Decision Tree

Accuracy: 0.8149626105054949

Precision, Recall, F1 (0.8116938702464295, 0.8075949367088607, 0.809024955319559)

---

Random Forest

Accuracy: 0.8117632424794261

Precision, Recall, F1 (0.8156512120388707, 0.8177215189873418, 0.8147570000318239)

---

Nearest Neighbors

Accuracy: 0.7494786750899238

Precision, Recall, F1 (0.7516183990912001, 0.7493670886075949, 0.7469975802567408)

---

OneR

Accuracy: 0.647916384398756

Precision, Recall, F1 (0.45988765649984104, 0.6481012658227848, 0.5275551477026753)

---

RBF SVM

Accuracy: 0.4863177567374958

Precision, Recall, F1 (0.23626982855311648, 0.4860759493670886, 0.31797813382787393)

---

ZeroR

Accuracy: 0.4860759493670886

---

Naive Bayes

Accuracy: 0.43583390169651626

Precision, Recall, F1 (0.5644487549593188, 0.43544303797468353, 0.4070404151316961)

-----