```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Reading the CSV file
student_per = pd.read_csv("/content/student-mat.csv",delimiter=";")
student_per
```


```
{"type":"dataframe","variable_name":"student_per"}
```


```python
student_per.shape #This dataset has 395 samples and 33 features.
```

```
(395, 33)
```

```python
student_per.info() #This gives the information about the columns of dataset, Using this we find all the non-numerical columns in dataset
#The given dataset doesn't have any null values.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
```

```
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

```
student_per.describe()
```

{"summary":"{\n \"name\": \"student_per\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 134.436252896189,\n   \"min\": 1.2760427246056245,\n   \"max\": 395.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   16.696202531645568,\n   17.0,\n   395.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Medu\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 138.80963938157987,\n   \"min\": 0.0,\n   \"max\": 395.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   395.0,\n   2.749367088607595,\n   3.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fedu\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 138.92085462409693,\n   \"min\": 0.0,\n   \"max\": 395.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   395.0,\n   2.5215189873417723,\n   3.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"traveltime\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 139.0946757987501,\n   \"min\": 0.6975047549086848,\n   \"max\": 395.0,\n   \"num_unique_values\": 6,\n   \"samples\": [\n   395.0,\n   1.4481012658227848,\n   4.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"studytime\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 139.00700274471274,\n   \"min\": 0.839240346418556,\n   \"max\": 395.0,\n   \"num_unique_values\": 6,\n   \"samples\": [\n   395.0,\n   2.0354430379746837,\n   4.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"failures\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 139.4513615014189,\n   \"min\": 0.0,\n   \"max\": 395.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.3341772151898734,\n   3.0,\n   0.743650973606249\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"famrel\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\":

```
138.45880901426744,\n        \"min\": 0.8966586076885056,\n
\"max\": 395.0,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          395.0,\n          3.9443037974683546,\n
5.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"freetime\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 138.63828826062982,\n        \"min\":
0.9988620396657205,\n        \"max\": 395.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          395.0,\n
3.2354430379746835,\n          4.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"goout\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 138.68948196584594,\
n        \"min\": 1.0,\n        \"max\": 395.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
3.108860759493671,\n          3.0,\n          395.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Dalc\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 139.0354623650101,\n
\"min\": 0.8907414280909659,\n        \"max\": 395.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          395.0,\n
1.481012658227848,\n          5.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Walc\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 138.87302263653973,\n
\"min\": 1.0,\n        \"max\": 395.0,\n        \"num_unique_values\":
7,\n        \"samples\": [\n          395.0,\n
2.2911392405063293,\n          3.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"health\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
138.50262599778412,\n        \"min\": 1.0,\n        \"max\": 395.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          395.0,\n
3.5544303797468353,\n          4.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"absences\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
136.85777166785417,\n        \"min\": 0.0,\n        \"max\": 395.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          395.0,\n
5.708860759493671,\n          8.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"G1\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 136.30663508587594,\n
\"min\": 3.0,\n        \"max\": 395.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          10.90886075949367,\n
11.0,\n          395.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"G2\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 136.4163745266465,\n        \"min\": 0.0,\n        \"max\":
```

```
395.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
10.713924050632912,\n          11.0,\n          395.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"G3\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 136.35024783099098,\n
\"min\": 0.0,\n        \"max\": 395.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          10.415189873417722,\n
11.0,\n          395.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

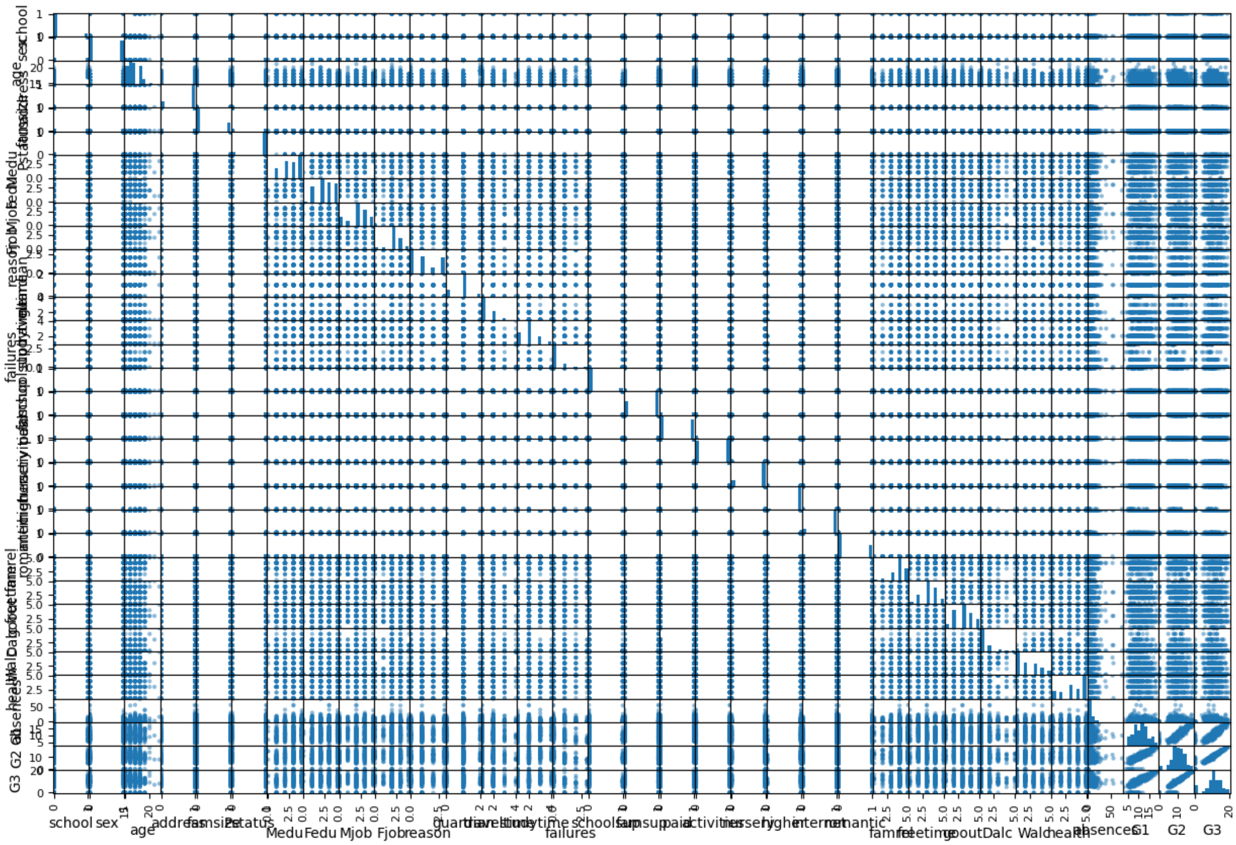

```
# Converting all the non-numerical columns in dataset to numerical
columns using Label Encoder
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
encode_colm = student_per.iloc[:,
[0,1,3,4,5,8,9,10,11,15,16,17,18,19,20,21,22]]
for i in encode_colm:
    student_per[i] = label.fit_transform(student_per[i])
student_per
```


```
{"type":"dataframe","variable_name":"student_per"}
```


```
# Correlation Matrix
correlation_Matrix = student_per.corr()
correlation_Matrix
```


```
{"type":"dataframe","variable_name":"correlation_Matrix"}
```


```
# Scatter Matrix
pd.plotting.scatter_matrix(student_per, figsize=(15, 10))
plt.show()
```




```
# Storing the input attributes and the target variable
X = student_per.iloc[:,:32] # Input attributes
y = student_per.iloc[:,32] # Target variable(G3)
X
```

{"type":"dataframe","variable_name":"X"}

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.20,random_state=42)

from sklearn.preprocessing import StandardScaler
stand_scale = StandardScaler()
stand_scale.fit(X_train)
X_train_std = stand_scale.transform(X_train)
X_test_std = stand_scale.transform(X_test)

from sklearn.linear_model import LinearRegression
LR_model = LinearRegression()
LR_model.fit(X_train_std,y_train)
LR_score = LR_model.score(X_test_std,y_test) # Finding the R2 score of
the model
print("R2 score:",LR_score)
```

```
R2 score: 0.7545777855043497
```

SUPPORT VECTOR MACHINES

```
from sklearn.svm import SVR
# SVM using rbf kernel
SVR_rbf = SVR(kernel='rbf')
# SVM using linear kernel
SVR_lin = SVR(kernel='linear')
# SVM using polynomial kernel
SVR_poly = SVR(kernel='poly')

SVR_rbf.fit(X_train_std,y_train)
SVR_lin.fit(X_train_std,y_train)
SVR_poly.fit(X_train_std,y_train)

# Finding the R2 score of SVR model using rbf kernel
rbf_score = SVR_rbf.score(X_test_std,y_test)
# Finding the R2 score of SVR model using linear kernel
lin_score = SVR_lin.score(X_test_std,y_test)
# Finding the R2 score of SVR model using polynomial kernel
poly_score = SVR_poly.score(X_test_std,y_test)

print("R2 score of SVM (rbf):",rbf_score)
print("R2 score of SVM (linear):",lin_score)
print("R2 score of SVM (polynomial):",poly_score)
```

```
R2 score of SVM (rbf): 0.6681792330768994
R2 score of SVM (linear): 0.7899692272521163
R2 score of SVM (polynomial): 0.49708883959770733
```

DECISION TREES

```
from sklearn.tree import DecisionTreeRegressor
tree_mod = DecisionTreeRegressor() # Performing DecisionTree Regressor model
tree_mod.fit(X_train,y_train)
tree_score = tree_mod.score(X_test,y_test) # Finding the R2 score
print("R2 score:",tree_score)
```

```
R2 score: 0.6845481824149033
```

```
# Visualising the Regression tree
feature_names = student_per.columns[:32]
from sklearn.tree import export_graphviz
regression_tree = export_graphviz(tree_mod,out_file="data\studentperformance.dot",feature_names=feature_names,rounded=True,filled=True)

from graphviz import Source
Source.from_file("data\studentperformance.dot")
```