STUDENT'S PERFORMANCE

By Group 9

G.Manasa
K.Akanksha
M.Swarna Lakshmi
T.Varsha

BACKGROUND

This data approach student achievement in secondary education of two Portuguese schools. The dataset is provided regarding the performance in Mathematics.

OBJECTIVE

Predict student performance G3 in secondary education(high school). To fit various models and compare the results.

THE PATH

Team followed a standard Machine Learning algorithm development process to predict the final grade G3.

ABOUT THE DATA

- The dataset has 33 attributes and 13035 observations.

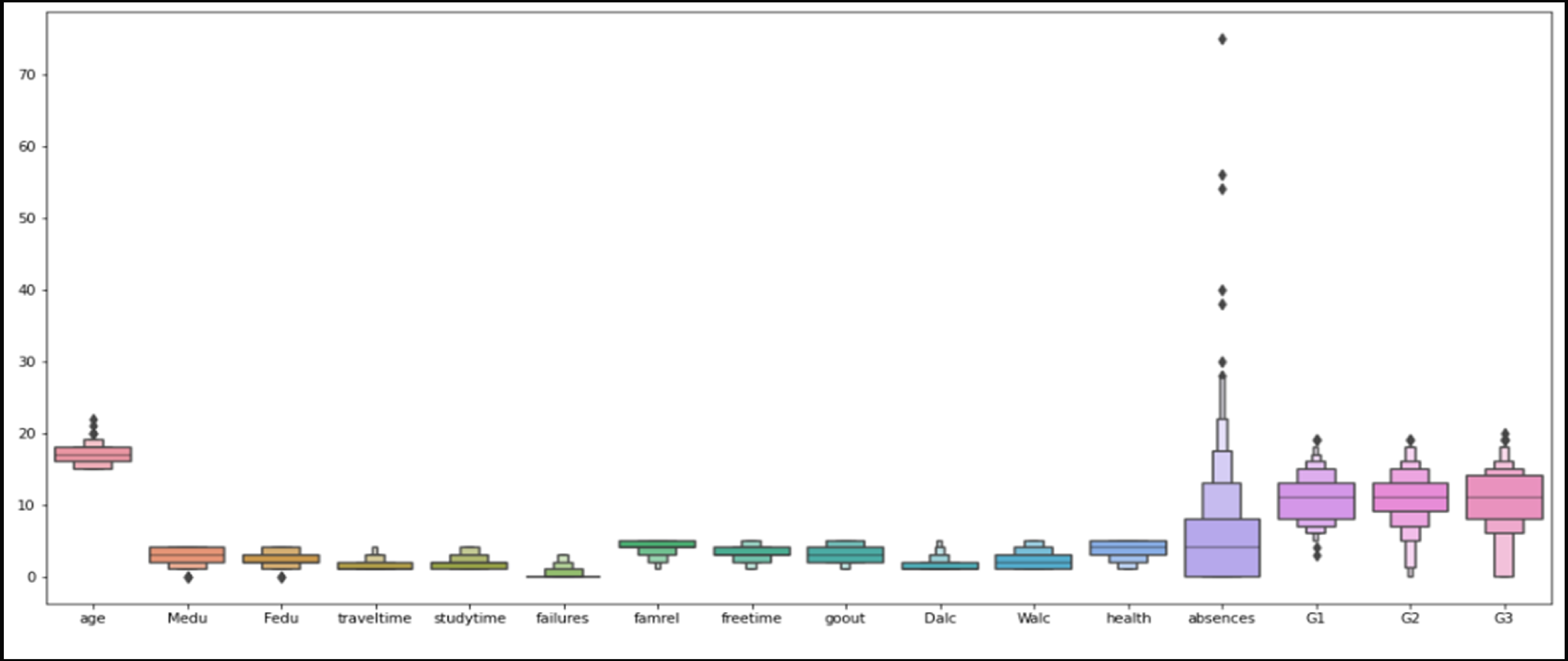
Categorical Variables		Continuous Variables	
School	Famsup	Age	Dalc
Sex	Paid	Medu	Walc
Address	Activities	Fedu	Health
Famsize	Nursery	Traveltime	Absences
Pstatus	Higher	Studytime	G1
Schoolsup	Internet	Failures	G2
Mjob	Romantic	Free time	G3
Fjob	Reason	Go out	

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10

Data Quality Check

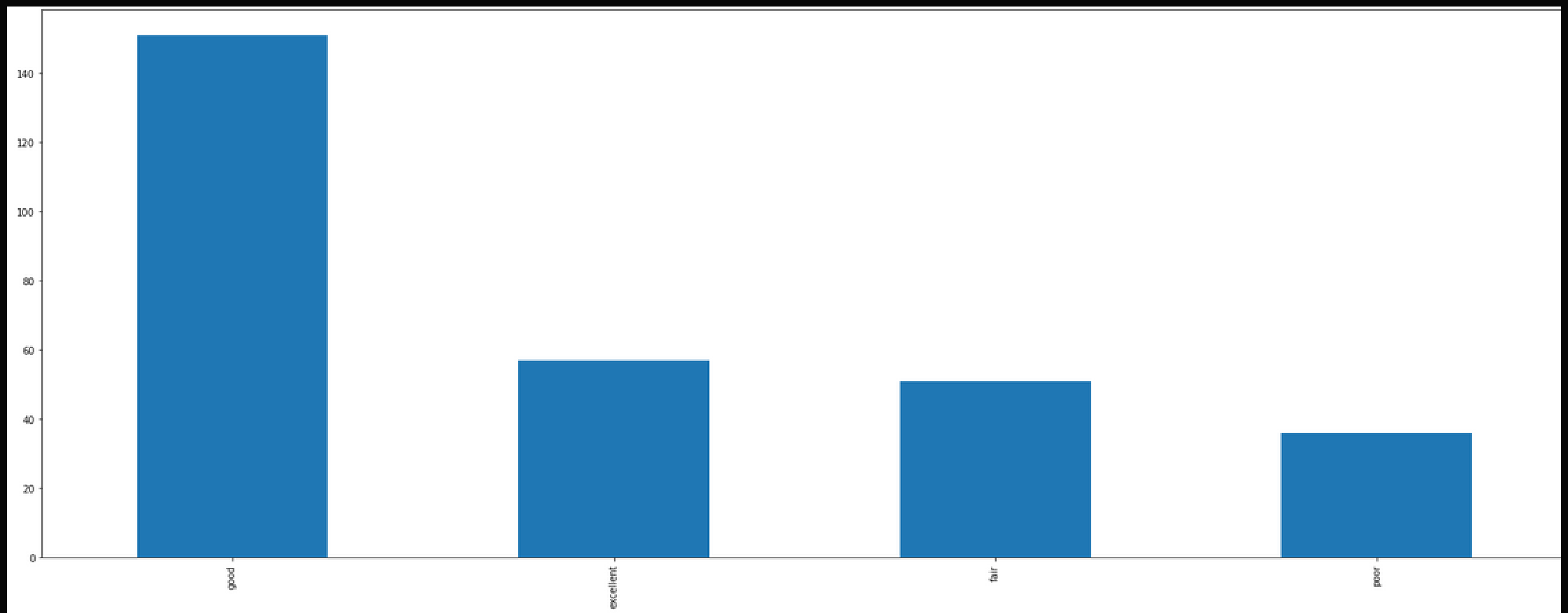
- **There are no missing values in the data**
- **We check for outliers in the data**

OUTLIERS



Outliers are found in age, Mother education, Father education,absences, G1,G2 and G3

EXPLORATORY DATA ANALYSIS:

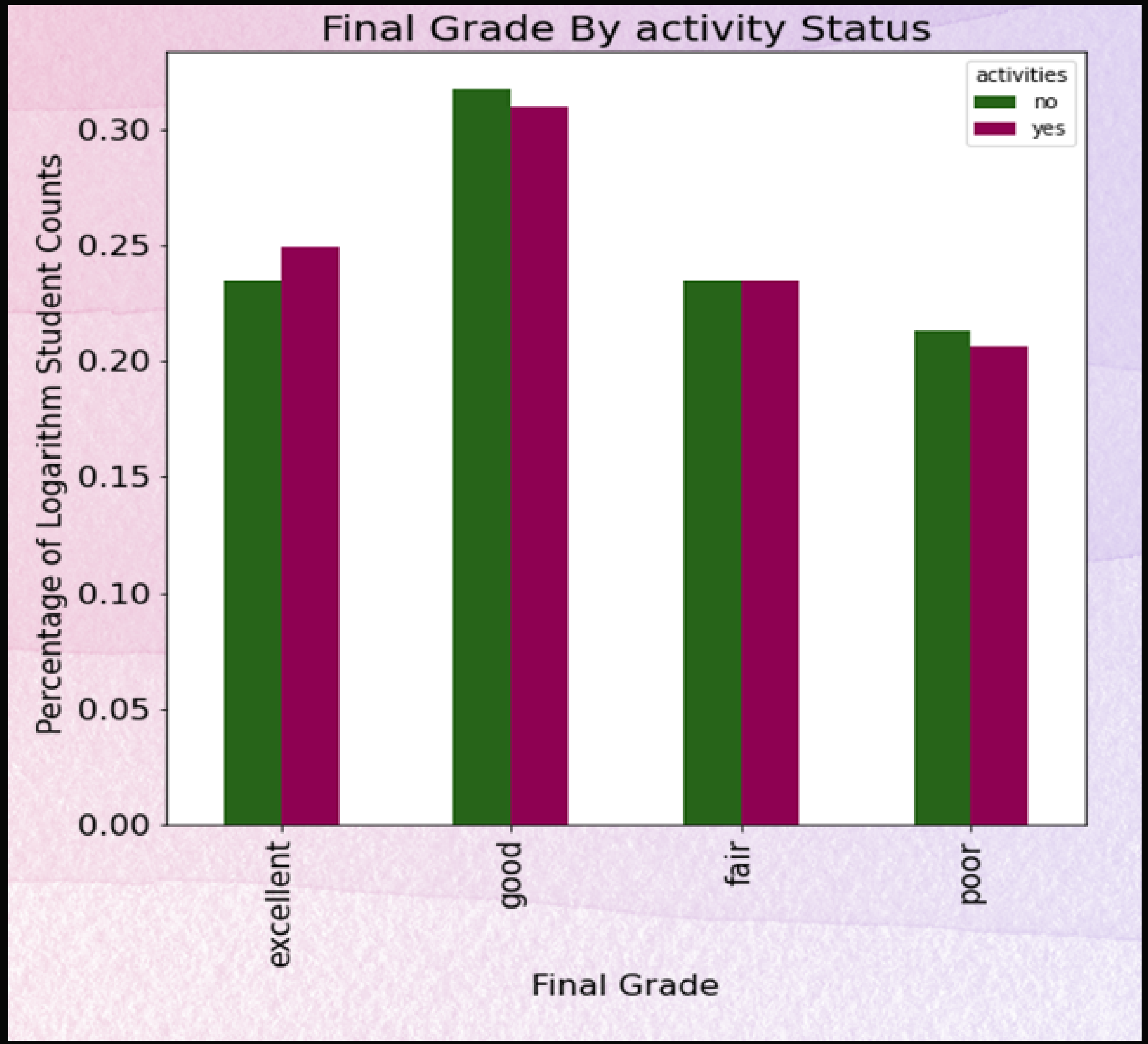


Good

Excellent

Fair

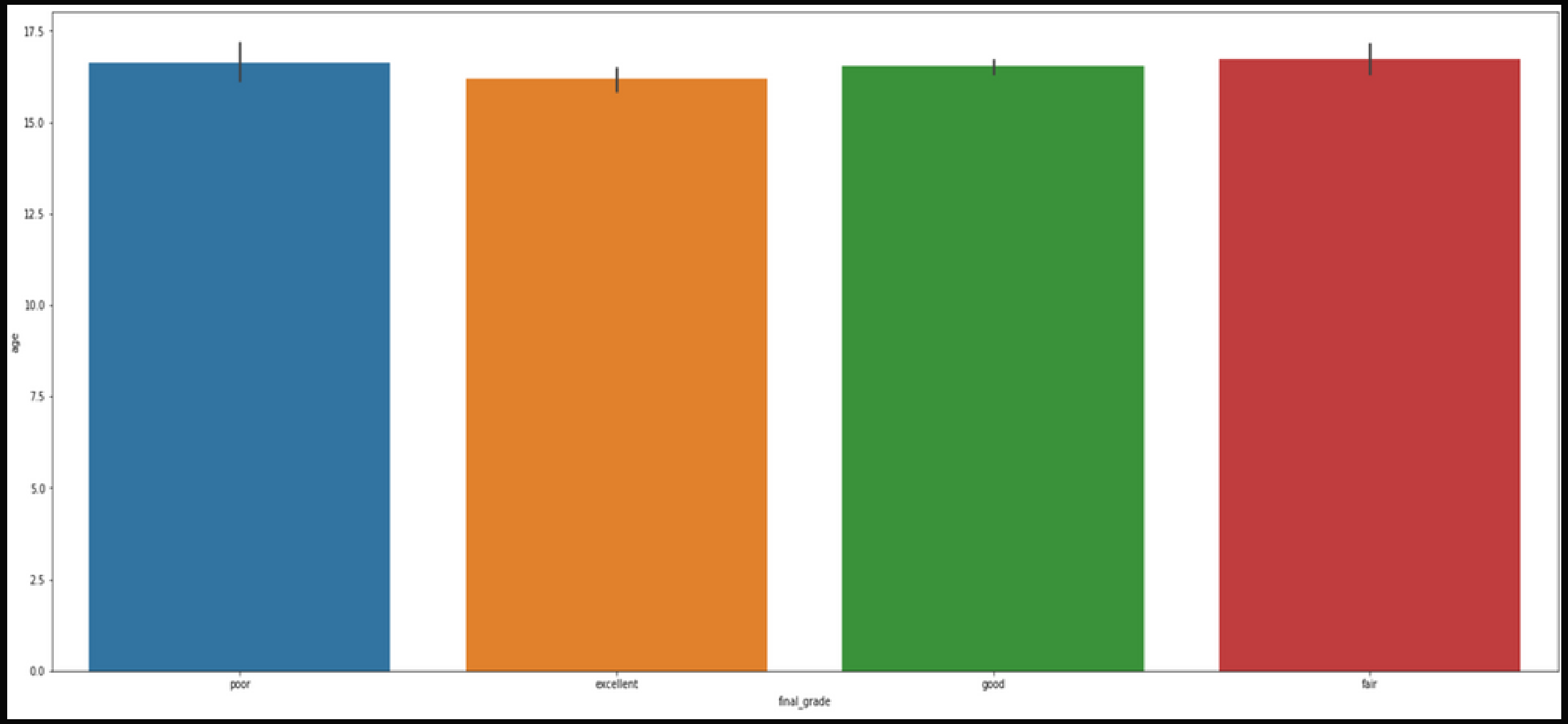
Poor



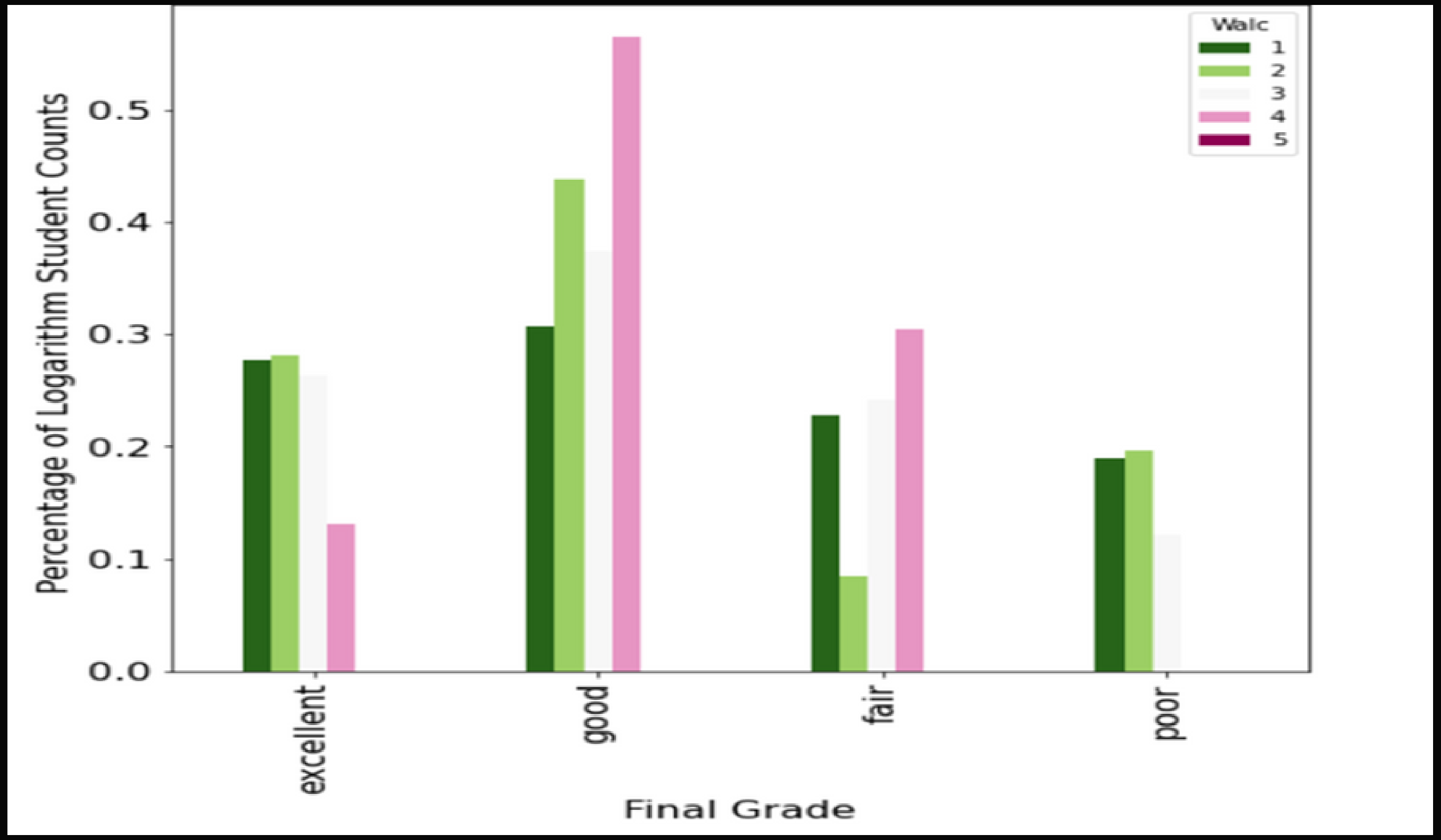
Student's Performance

Activity Status vs Grade

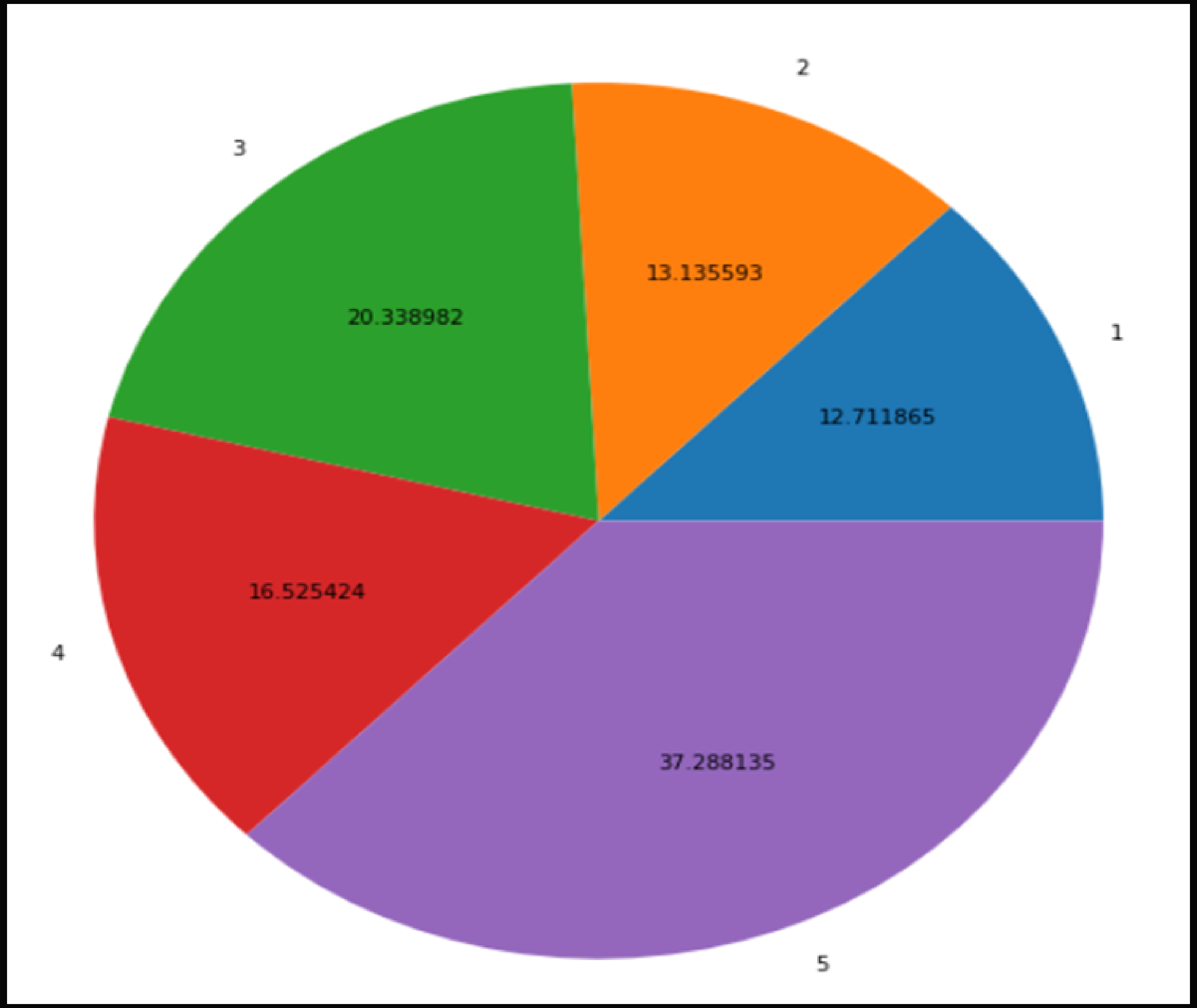
<u>**Age vs Grade**</u>

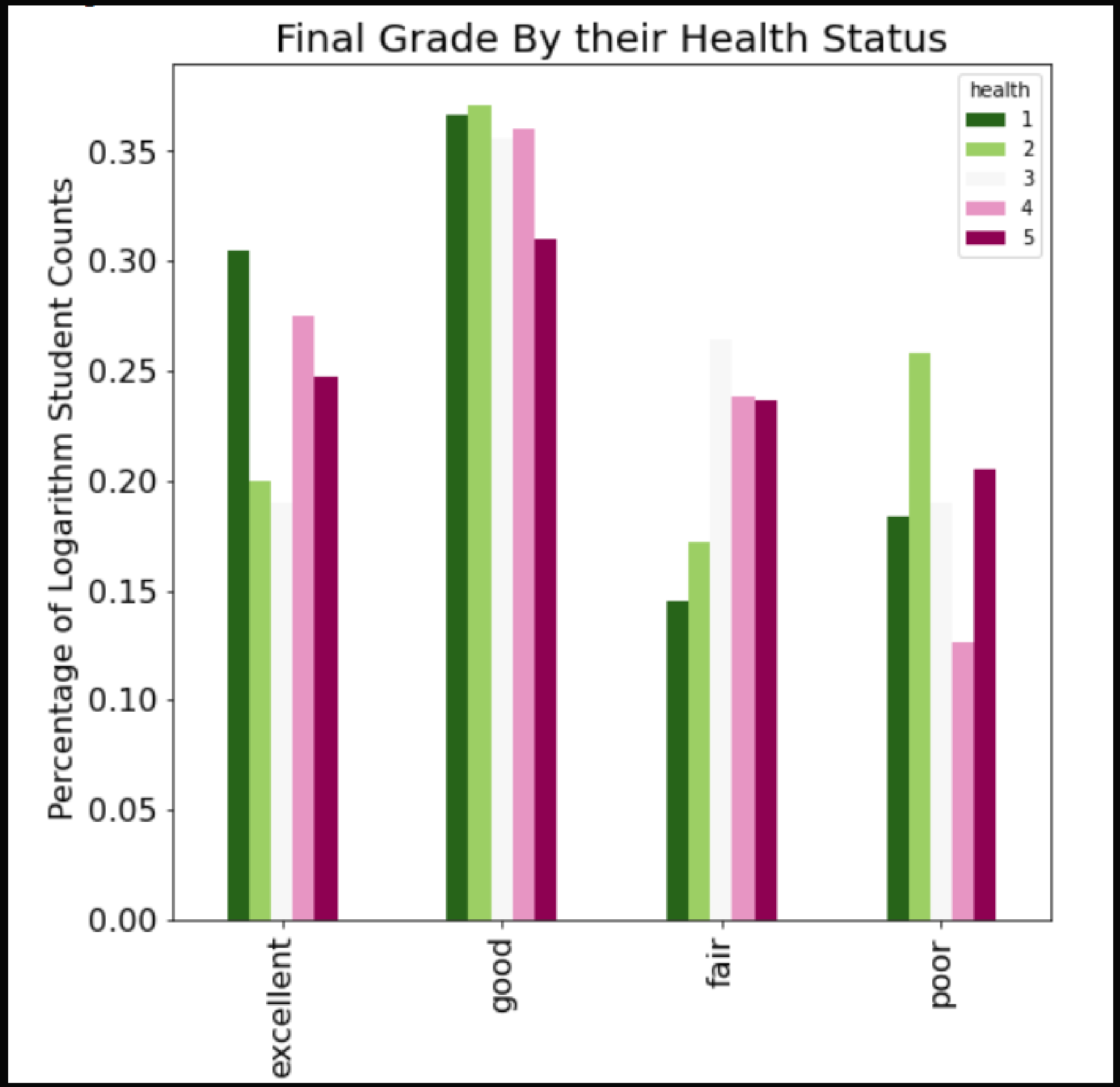


<u>**Alcohol vs Grade**</u>

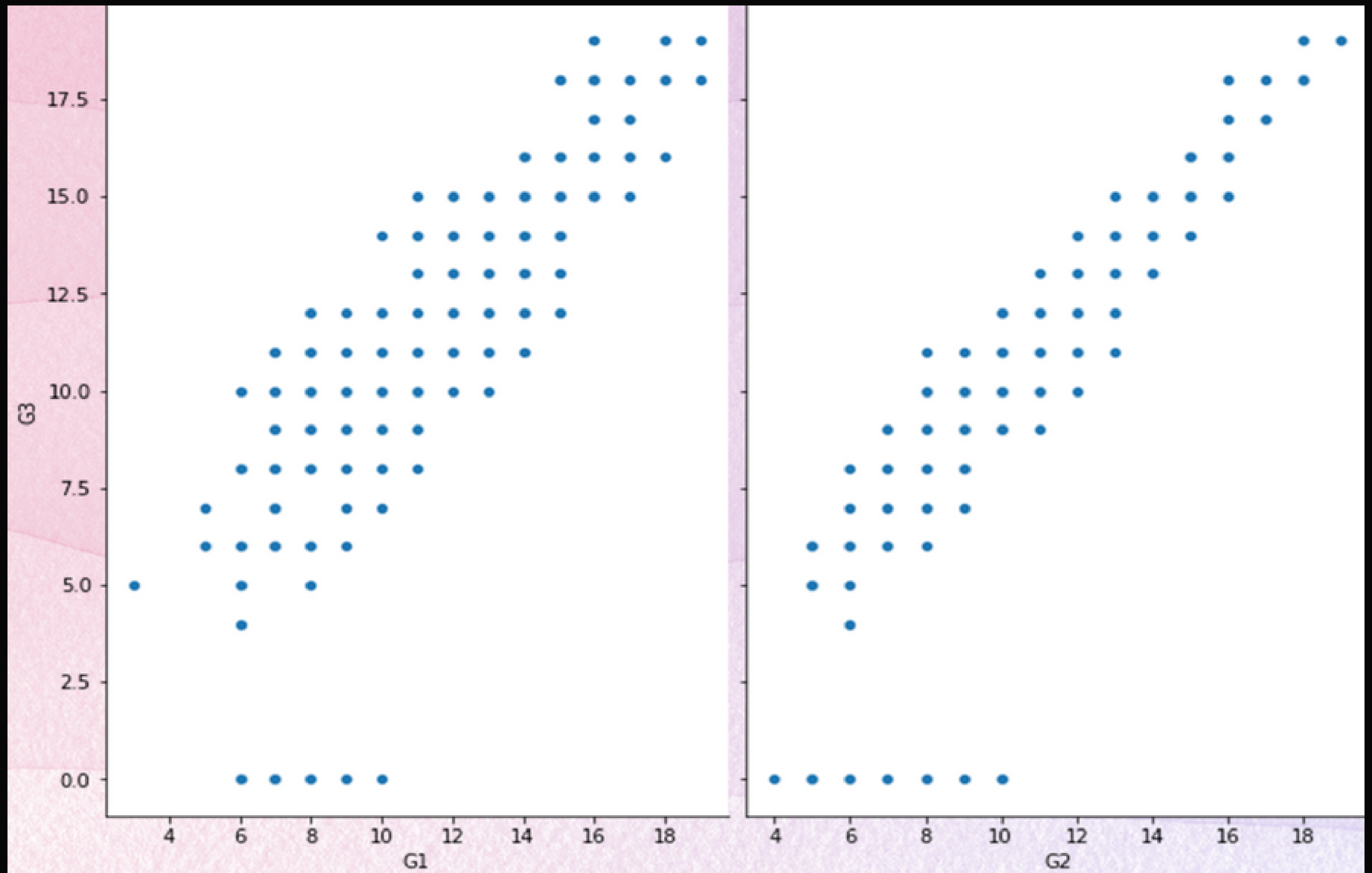


Health Status

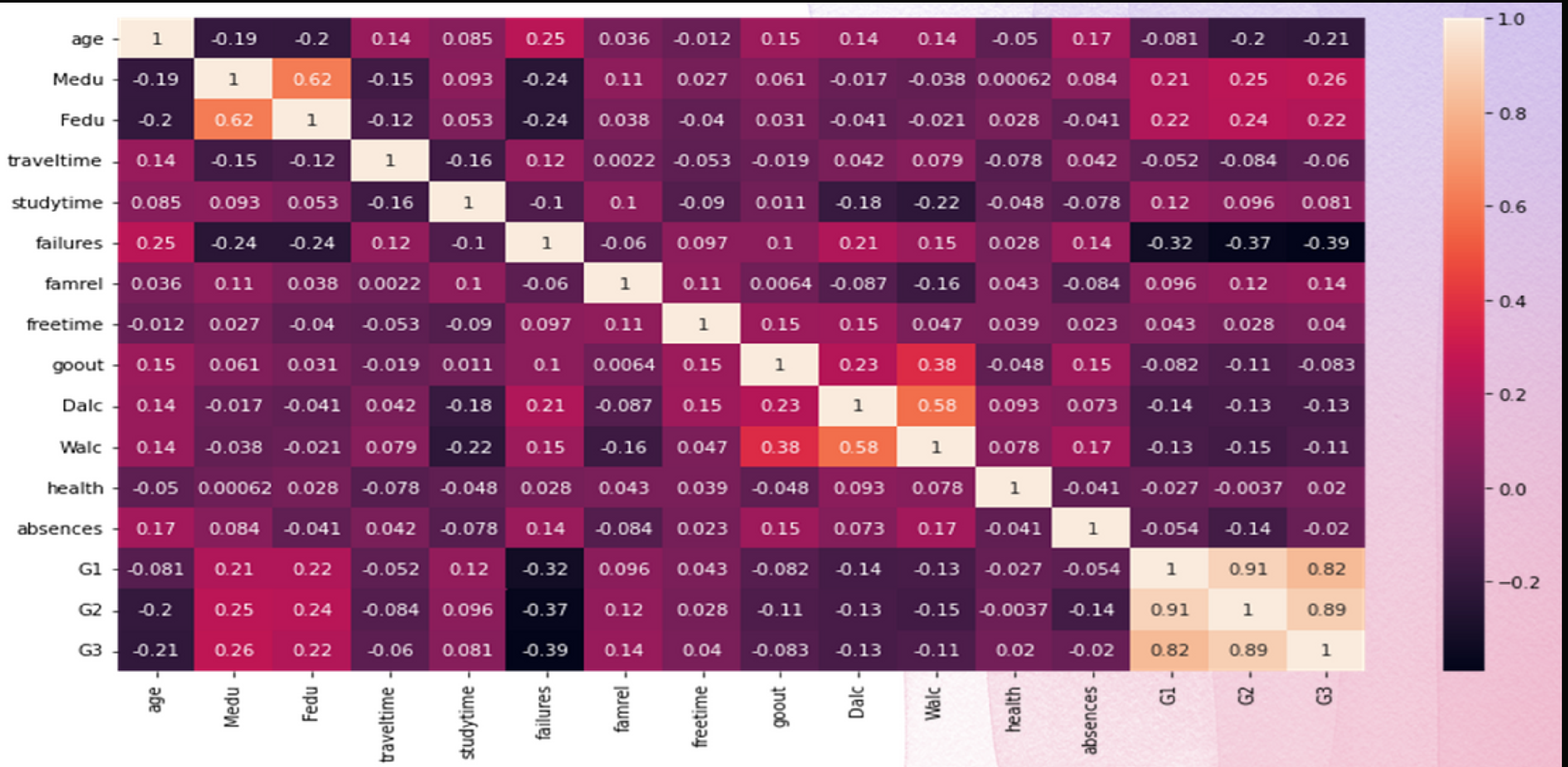




PAIRPLOT



MULTICOLLINEARITY CHECK



LINEAR REGRESSION

MODEL 1		
Train-test	Accuracy	MAE
80-20	0.83	0.98
75-25	0.82	1.004
70-30	0.85	1.07
60-40	0.84	1.11

In model 1, we dropped few variables based on the heatmap and calculated mean absolute error for various train-test splits.

The mean absolute error for 80-20 ratio is 0.98 with accuracy of 83%

Since all the features were not considered in the previous model, we again fit the model considering all the features with various train-test ratios.

MODEL 2		
Train-Test	Accuracy	MAE
80-20	0.867	1.04
75-25	0.86	1.06
70-30	0.82	1.26
60-40	0.80	1.37

The least mean absolute error we get for the model 2 is 1.04 i.e for 80-20 train test split with 87% accuracy.

NEURAL NETWORKS

- For fitting a neural model, we consider 'X' as an independent variable that takes all the features except the feature that we are going to predict i.e G3 which is considered as dependent varaible 'y'.

- We have tried taking different train-test ratios with various architectures to get an minimized error.

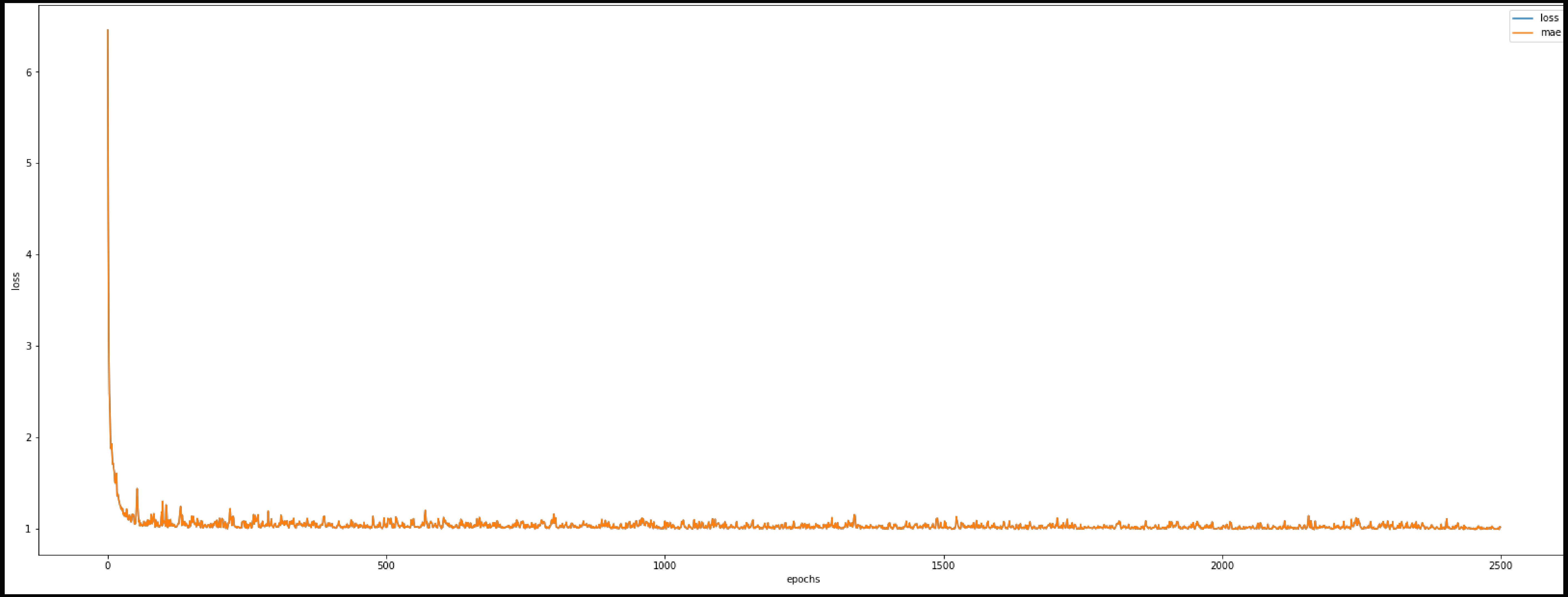
- The optimizers used for neural network are 'SGD' and 'Adam'.

TRAIN-TEST	Architecture	Epochs	Optimizer	MAE		Optimizer	MAE
70-30	28-14-7-3-1	300	Adam	1.0705		SGD	1.141
70-30	28-14-7-3-1	500	Adam	1.0475		SGD	1.1587
70-30	28-14-7-3-2	800	Adam	1.0316		SGD	1.1554
70-30	28-14-7-3-1	1500	Adam	1.027		SGD	1.2489
70-30	28-14-7-3-1	2200	Adam	1.0263		SGD	1.0826
80-20	28-14-7-3-2-1	200	Adam	0.9819		SGD	1.7815
80-20	28-14-7-3-2-1	700	Adam	1.0736		SGD	1.1993
80-20	28-14-7-3-2-1	1200	Adam	1.1146		SGD	1.2894
80-20	28-14-7-3-2-1	2000	Adam	0.9725		SGD	1.4479
80-20	28-14-7-3-2-1	2500	Adam	0.9590		SGD	1.2821
75-25	14-7-3-2-1	600	Adam	0.964		SGD	1.1199
75-25	14-7-3-2-1	1100	Adam	1.125		SGD	1.2552
75-25	14-7-3-2-1	1800	Adam	0.911		SGD	1.173
75-25	14-7-3-2-1	2300	Adam	1.0013		SGD	1.1622
75-25	14-7-3-2-1	3000	Adam	0.9083		SGD	1.2735
60-40	14-7-01	300	Adam	1.1353		SGD	1.0805
60-40	14-7-01	600	Adam	1.0445		SGD	1.184
60-40	14-7-01	1350	Adam	1.0461		SGD	1.1604
60-40	14-7-01	1700	Adam	1.1756		SGD	1.2481
60-40	14-7-01	2200	Adam	1.0415		SGD	1.2578

Optimizer: Adam

Architecture: 28-14-7-3-2-1

epochs: 2500



BAGGING

Bootstrap

MAE:0.89

Accuracy:0.90

Random Forest

MAE: 0.82

Accuracy:0.92

Decision Trees

MAE:1.07

Accuracy:0.88

Random Forest

MAE : 0.82

Least Among Bagging

BOOSTING

Adaptive Boosting

MAE: 0.96

Accuracy:0.88

Gradient Boosting

MAE: 1.07

Accuracy:0.87

XG Boosting

MAE: 0.83

Accuracy:0.92

Least Among Boosting

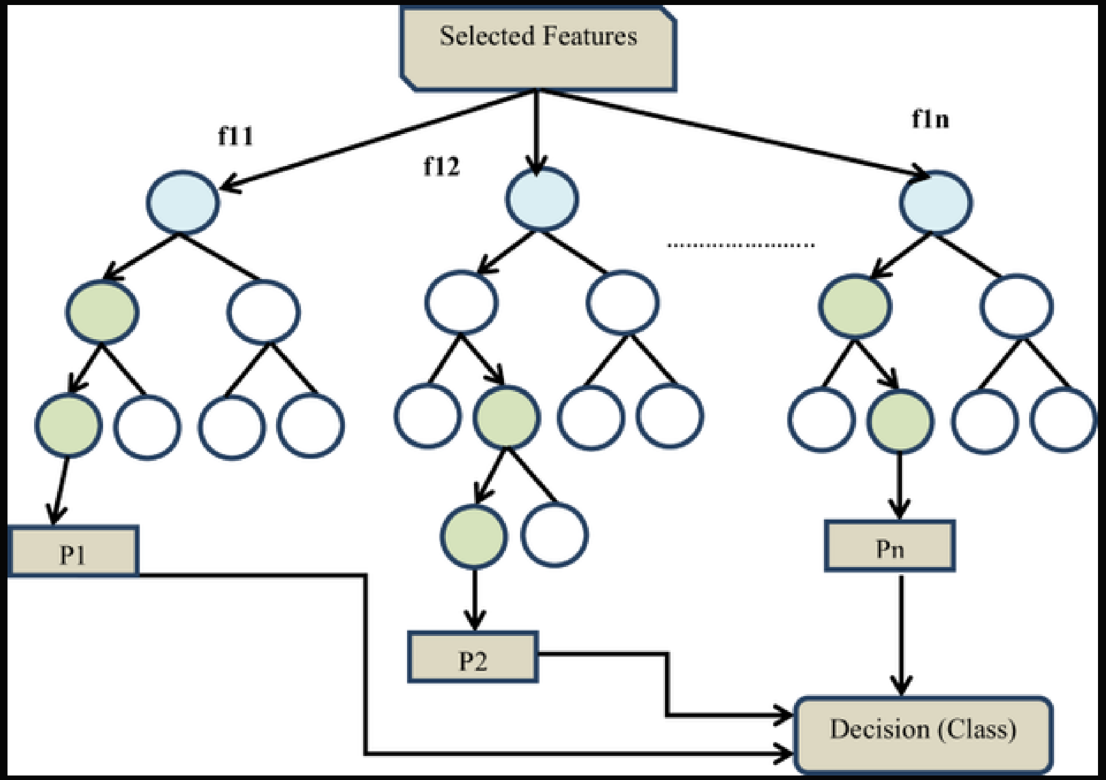
XG Boosting

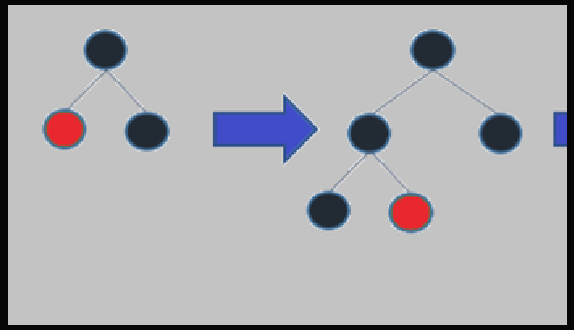
MAE: 0.83

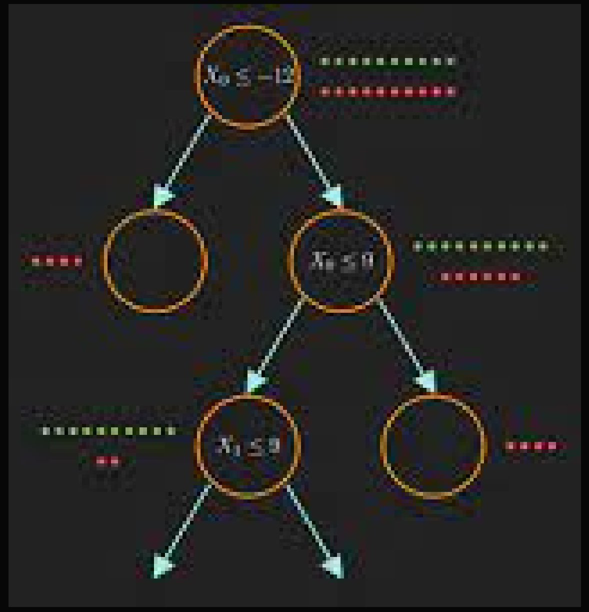
CONCLUSION

Model	Accuracy	MAE
Random Forest	0.92	0.82
XG Boosting	0.92	0.83
Bootstrap	0.90	0.89
Neural Networks	0.80	0.96
Adaptive Boosting	0.88	0.96
Linear Regression	0.87	1.04
Decision Trees	0.88	1.07
Gradient Boosting	0.87	1.07

The least mean absolute error for this dataset when compared to all the models is for Random Forest i.e 0.82







The highest mean absolute error for this dataset when compared to all the models is for Decision Trees and Gradient Boosting i.e 1.07.

Presented by

G.Manasa

K.Akanksha

M.Swarna Lakshmi

T.Varsha

References

THANK YOU

ATTRIBUTE DESCRIPTION

- School - student's school
- Sex - student's sex
- Age - student's age
- Address - student's home address type
- Famsize - family size
- Pstatus - parent's cohabitation status
- Medu - mother's education
- Fedu - father's education
- Mjob - mother's job
- Fjob - father's job
- Reason - reason to choose this school
- Guardian - student's guardian
- Traveltime - home to school travel time

- Studytime - weekly study time
- Failures - number of past class failures
- Schoolsup - extra educational support
- Activities - extra-curricular activities
- Nursery - attended nursery school
- Higher - wants to take higher education
- Internet - Internet access at home
- Romantic - with a romantic relationship
- Famrel - quality of family relationships
- Freetime - free time after school
- Goout - going out with friends
- Dalc - workday alcohol consumption
- Walc - weekend alcohol consumption

Health - current health status

- Absences - number of school absences
- G1 - first period grade
- G2 - second period grade
- G3 - final grade
- Famsup - family educational support
- Paid - extra paid classes within the course subject

OUTLIER TREATMENT

We have written a function to identify the outliers that returns a logic which gives us the rows having outliers .

```
plt.rcParams["figure.figsize"] = [18.50, 8.50]
out=sns.boxenplot(data=df)
```

REMOVAL OF OUTLIERS

```
for x in ['absences']:
    q75,q25 = np.percentile(df.loc[:,x],[75,25])
    intr_qr = q75-q25

    max = q75+(3.5*intr_qr)
    min = q25-(3.5*intr_qr)

    df.loc[df[x] < min,x] = np.nan
    df.loc[df[x] > max,x] = np.nan
```

```
for x in ['Fedu','traveltime','studytime','failures','famrel','freetime','Dalc','age','G1','G2','G3']:
    q75,q25 = np.percentile(df.loc[:,x],[75,25])
    intr_qr = q75-q25

    max = q75+(1.5*intr_qr)
    min = q25-(1.5*intr_qr)

    df.loc[df[x] < min,x] = np.nan
    df.loc[df[x] > max,x] = np.nan
```

```
df.dropna(inplace=True)
```

LINEAR REGRESSION

```
x=df.drop(['G3'],axis=1)
print(x)
y=df['G3']
print(y)
```

```
from sklearn.linear_model import LinearRegression
import statsmodels.formula.api as smf
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score

linreg = LinearRegression()
linreg.fit(X, y)
df['G3_pred']=linreg.predict(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0,train_size=0.80)
lm2 = LinearRegression()
lm2.fit(X_train, y_train)
y_pred = lm2.predict(X_test)

skl.mean_absolute_error(y_test,y_pred)
```

```
1.0353287547638819
```

```
[1011] import tensorflow as tf
```

```
[1021] tf.random.set_seed(42)
```

```
[1022] model= tf.keras.Sequential([
                             tf.keras.layers.Dense(28),
                            tf.keras.layers.Dense(14),
                            tf.keras.layers.Dense(7),
                             tf.keras.layers.Dense(3),
                             tf.keras.layers.Dense(2),
                            tf.keras.layers.Dense(1)
       ])
```

```
[1023] model.compile(loss= tf.keras.losses.mae,
              optimizer= tf.keras.optimizers.Adam(),
              metrics= ["mae"])
```

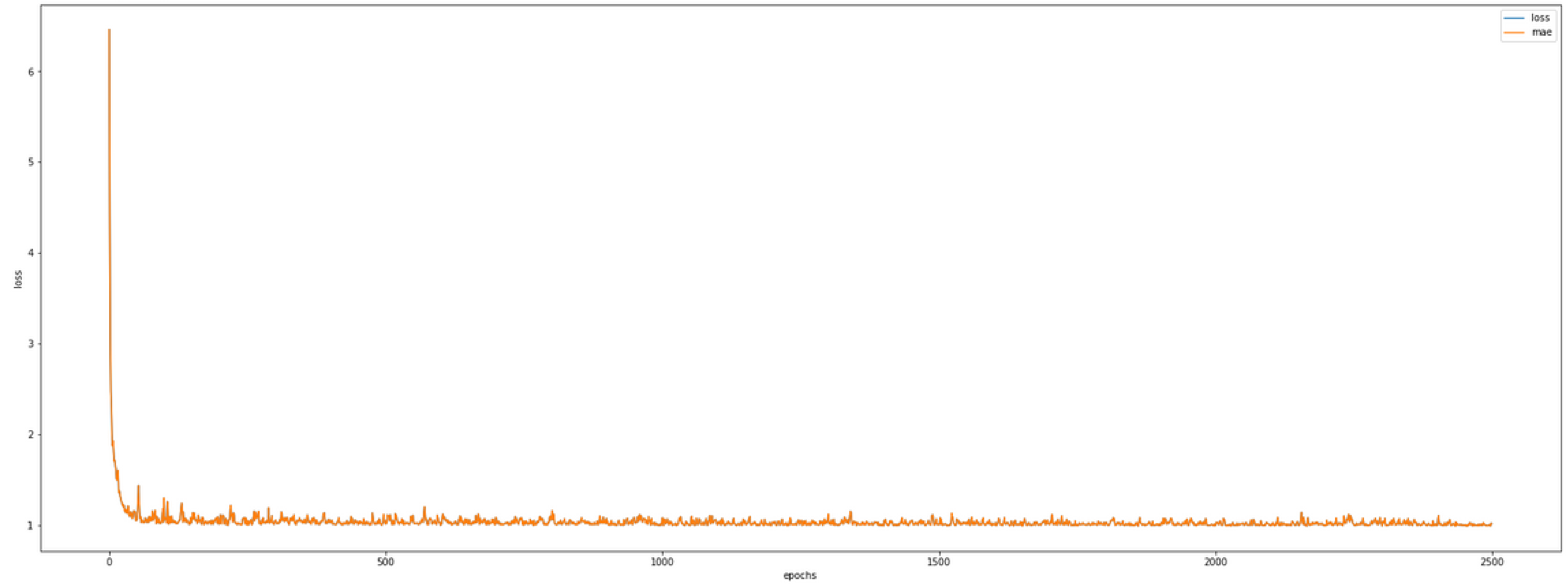
```
[1024] history= model.fit(X_train, y_train, epochs=2500, verbose=0)
```

```
[1025] model.evaluate(X_test, y_test)

       2/2 [==============================] - 0s 8ms/step - loss: 0.9590 - mae: 0.9590
       [0.9589898586273193, 0.9589898586273193]
```

```
pd.DataFrame(history.history).plot()
plt.ylabel("loss")
plt.xlabel("epochs")

Text(0.5, 0, 'epochs')
```




```
[336] from sklearn.metrics import r2_score
      y_pred=model.predict(X_test)
      r2=metrics.r2_score(y_test,y_pred)
      r2

      0.8043843288849539
```

BOOTSTRAP

```
from sklearn.ensemble import BaggingRegressor
import sklearn.metrics as metrics
bag_model = BaggingRegressor(
base_estimator=BaggingRegressor(),
n_estimators=100,
max_samples=0.8,
bootstrap=True,
oob_score=True,
random_state=42
)
l=bag_model.fit(X_train, y_train)
```

```
mae = metrics.mean_absolute_error(y_test, l.predict(X_test))

print("The mean abs error (MAE) on test set: {:.4f}".format(mae))
```

```
The mean abs error (MAE) on test set: 0.8957
```

```
l.score(X_test,y_test)
```

```
0.9043706615762115
```

RANDOM FOREST

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=100, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 0.8245762711864406
```

```
regressor.score(X_test,y_test)
```

```
0.9294536354244018
```

DECISION TREES

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(max_depth=3,min_samples_leaf = 10, random_state = 42)
regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(max_depth=3, min_samples_leaf=10, random_state=42)
```

```
y_pred = regressor.predict(X_test)
```

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 1.0701844647582062
```

```
regressor.score(X_test,y_test)
```

```
0.8801873344304388
```

```
df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
df
```

	Actual	Predicted
313	11.0	8.000000
311	13.0	11.833333
13	11.0	10.059701
291	15.0	15.392857
355	9.0	10.059701
53	11.0	10.059701
323	15.0	13.513514
72	5.0	7.194444
289	15.0	13.513514
84	10.0	10.059701
394	9.0	10.059701
16	14.0	13.513514
200	16.0	15.392857
305	12.0	11.833333

ADAPATIVE BOOSTING

```
#Adaboosting
from sklearn.ensemble import AdaBoostRegressor

adaclf = AdaBoostRegressor(
                             n_estimators=100,
                             learning_rate=0.1,
                             random_state=42)
adaclf.fit(X_train,y_train)
y_pred_1 = adaclf.predict(X_test)
ab=mean_absolute_error(y_test, y_pred_1)
print(ab)
```

```
0.959795807898721
```

```
adaclf.score(X_test,y_test)
```

```
0.8829311834247204
```

GRADIENT BOOSTING

```
from sklearn import datasets, ensemble
from sklearn.inspection import permutation_importance
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error
```

```
#Gradient boosting
regressor1 = GradientBoostingRegressor(max_depth=4,n_estimators=15,learning_rate=0.1,random_state=0)
regressor1.fit(X_train, y_train)
```

```
GradientBoostingRegressor(max_depth=4, n_estimators=15, random_state=0)
```

```
y_pred = regressor.predict(X_test)
mean_absolute_error(y_test, y_pred)
```

```
1.0701844647582062
```

```
regressor1.score(X_test,y_test)
```

```
0.872246643737185
```

EXTREME GRADIENT BOOSTING

```
#ExtremeGradient Boosting
from xgboost import XGBRegressor
clf = XGBRegressor(n_estimators=100,
                   learning_rate=0.1,
                   random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
eg=mean_absolute_error(y_test, y_pred)
print(eg)
```

```
[15:52:02] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
0.8329001584295499
```

```
clf.score(X_test,y_test)
```

```
0.9294409258725392
```

The End