In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [ ]:
```
df_student = pd.read_csv("StudentScore.xls")
df_student.sample(10)
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
233	male	group E	some high school	standard	none	92	87	78
673	female	group C	associate's degree	standard	completed	65	84	84
96	male	group B	some high school	standard	completed	65	66	62
761	female	group D	some high school	standard	none	48	58	54
476	male	group E	bachelor's degree	standard	completed	76	62	66
170	male	group A	high school	standard	completed	72	73	74
379	male	group A	bachelor's degree	standard	none	66	64	62
521	female	group C	associate's degree	standard	none	91	86	84
209	female	group B	some college	free/reduced	none	58	61	66
868	male	group E	associate's degree	free/reduced	completed	78	74	72

check these value in the each of columns

In [ ]:
```
print(df_student["gender"].unique())
print(df_student["race/ethnicity"].unique())
print(df_student["parental level of education"].unique())
print(df_student["lunch"].unique())
print(df_student["test preparation course"].unique())
```

```
['female' 'male']
['group B' 'group C' 'group A' 'group D' 'group E']
["bachelor's degree" 'some college' "master's degree" "associate's degree"
 'high school' 'some high school']
['standard' 'free/reduced']
['none' 'completed']
```

i want to predict "math score" of student so i set value of target variabel is math score and next step, i split the data to train, test data

In [ ]:
```
target = "math score"
X = df_student.drop(target, axis=1)
y = df_student[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

Data Prepreocessing

- Numercial : reading score, writting score
- Catergorical:
  - Norminal : gender, race/ethnicity, lunch, test preparation course
  - Ordinal : parental level of education
  - Boolean :
- The knowlegde necessity

Use Pipline: Pipeline

columns transformer: Columns Transformer

lazypredict this here

In [ ]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OrdinalEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.svm import SVR
from lazypredict.Supervised import LazyRegressor
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor



education_level = ["some high school", "high school", "some college", "associate's
gender_level = ["female", "male"]
lunch_level = df_student["lunch"].unique()
test_preparation = df_student["test preparation course"].unique()

numercial_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

Ordinal_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('scaler', OrdinalEncoder(categories=[education_level, gender_level, lunch_leve
])
Norminal_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('scaler', OneHotEncoder(sparse_output=False))
])

preprocesser = ColumnTransformer(transformers=[
    ('num_features', numercial_transformer, ["reading score", "writing score"] ),
    ('ordinal_features', Ordinal_transformer,["parental level of education", "gende
    ('num_feature', Norminal_transformer, ["race/ethnicity"])
] )

reg = Pipeline(steps=[
    ("preprocessing", preprocesser),
    ("model", RandomForestRegressor() )
])

# reg.fit(X_train, y_train)
# y_predict = reg.predict(X_test)
```

```
# # for i, j in zip(y_predict, y_test):
# #     print("Predict: {}. Actual: {}".format(i, j))
# print("MSE {}".format(mean_squared_error(y_test, y_predict)))
# print("MAE {}".format(mean_absolute_error(y_test, y_predict)))
# print("R2 {}".format(r2_score(y_test, y_predict)))

# Random forest
# MSE 35.23067096333333
# MAE 4.798896666666667
# R2 0.8299907579402186

# reg = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric=None)
# models, predictions = reg.fit(X_train, X_test, y_train, y_test)
# print(models)

params = {
    "model__n_estimators": [50, 100, 200],
    "model__criterion": ["squared_error", "absolute_error", "fried_mse", "possion"]
    "model__max_features": ["sqrt", "log2", None]
    # "preprocessing__num_features__imputer__strategy":
}


# grid_reg = GridSearchCV(reg, param_grid=params, verbose= 2, scoring="r2", n_jobs=
grid_reg = RandomizedSearchCV(reg, param_distributions=params, verbose= 2, scoring=
grid_reg.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

Out[ ]: