PROBLEM STATEMENT

A client's requirement is, he wants to predict the concrete strength based on the several parameters. The Client has provided the dataset of the same. As a data scientist, you have to develop a predictive model which will predict the concrete strength.

1.) Identify your problem statement

2.) Tell basic info about the dataset (Total number of rows, columns)

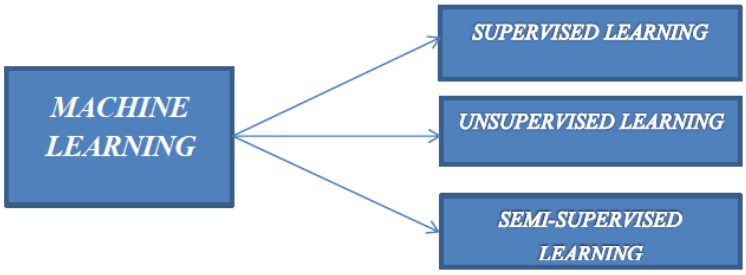
3.) Mention the pre-processing method if you're doing any (like converting string to number – nominal data)

4.) Develop a good model with good evaluation metric. You can use any machine learning algorithm; you can create many models. Finally, you have to come up with final model.

5.) All the research values of each algorithm should be documented. (You can make tabulation or screenshot of the results.)

6.) Mention your final model, justify why u have chosen the same.

Algorithm Identification



1) In machine learning, the requirement should be clear in the given dataset. Input and output are well defined.

Then it is **SUPERVISED LEARNING**

2) Under supervised learning, there are 2 types: Classification and Regression.

i)Classification - It has categorical
ii)Regression - It has numerical

An output parameter has numerical value, then it is REGRESSION

3)In regression algorithm, using **Multiple linear regression, Support vector machine, Decision tree and Random forest** on the model

Multiple Linear Regression

A)**Multiple Linear Regression –**Multiple linear regression is used to model the relationship between a continuous response variable and continuous or categorical explanatory variables. Recall that simple linear regression can be used to predict the value of a response based on the value of one continuous predictor variable.

MATPLOT-(**Multiple Linear Regression**)

1. Importing the Libraries - (input)

```
import numpy as np # numpy is numerical
import matplotlib.pyplot as plt # matplotlip is a plot the points in graph
import pandas as pd #used for data analysis
```

2. Loaded the excel into Jupyter Notebook - (input)

```
# Reading the Dataset
dataset = pd.read_csv('Concrete_Data_Yeh.csv')
```

3. Analysing & Visualization the given Data - (output)

```
]: dataset
```

]:

	cement	slag	flyash	water	superplasticizer	coarseaggregate	fineaggregate	age	csMPa
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
...	...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.18
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.70
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.77
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40

1030 rows × 9 columns

4. Create the dummies Bec string value is can't find ml authorisms

(input) (eg):Help to create dataset

```
: dataset=pd.get_dummies(dataset,drop_first=True)
```

```
: dataset
```

(Output)

```
: dataset
```

:

	cement	slag	flyash	water	superplasticizer	coarseaggregate	fineaggregate	age	csMPa
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
...	...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.18
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.70
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.77
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40

1030 rows × 9 columns

5. Separate the Dataset input and output column

Input column

- **Cement**
- **Slag**
- **Flyash**
- **Water**
- **Superplasticizer**
- **Coarseaggregate**
- **Fineaggregate**
- **age**

Output column

- CsMPa

(input)

```
indep=dataset[['cement', 'slag', 'flyash','water', 'superplasticizer','coarseaggregate','fineaggregate','age']]
dep=dataset['csMPa']
```

a)Independent Input

(Output)

```
indep
```

	cement	slag	flyash	water	superplasticizer	coarseaggregate	fineaggregate	age
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360
...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28

1030 rows × 8 columns

b)Dependent Output

(Output)

```
dep
```

```
0       79.99
1       61.89
2       40.27
3       41.05
4       44.30
        ...
1025    44.28
1026    31.18
1027    23.70
1028    32.77
1029    32.40
Name: csMPa, Length: 1030, dtype: float64
```

6. Split the Dataset 8% training data 2% test data

```
#split into training set and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(indep, dep, test_size = 1/3, random_state = 0)
```

7.**linear regression model & formula** **eg:** y=W1*w2*w3*w4*w5*x1+b0 for this equation we got value for b1 and bo

(input)

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```

7. Visualization the weight and bias value

(input)

```
weight=regressor.coef_
print("Weight of the model={}".format(weight))
bais=regressor.intercept_
print("Intercept of the model={}".format(bais))
```

(Output)

```
Weight of the model=[ 0.1244788   0.10816318  0.0940972  -0.11776245  0.38679612  0.0243998
  0.02529243  0.10774492]
Intercept of the model=-41.80447230512486
```

8. Prediction

(input)/ (Output)

```
cement_input=float(input("cement:"))
slag_input=float(input("slag:"))
flyash_input=float(input("flyash:"))
water_input=float(input("water:"))
superplasticizer_input=float(input("superplasticizer:"))
coarseaggregate_input=float(input("coarseaggregate:"))
fineaggregate_input=float(input("fineaggregate:"))
age_input=float(input("age:"))
```

```
cement:540
slag:0
flyash:0
water:162
superplasticizer:2.5
coarseaggregate:1040
fineaggregate:676
age:28
```

9. Future Prediction

(input)/ (Output)

```
Future_Prediction=regressor.predict([[cement_input,slag_input,flyash_input,water_input,superplasticizer_input,coarseaggregate_inp
print("Future_Prediction={}".format(Future_Prediction))
```

```
Future_Prediction=[52.79388158]
```

10. Output of the Training Dataset we are predicting the Value for the Test Dataset

(input)/ (Output)

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

```
Y_pred
```

```
array([ 4.41268512e+03,  2.77584291e+04,  4.19182212e+03, -7.77682840e+02,
        3.30229640e+04,  1.30664021e+04,  3.60458425e+04,  1.03474537e+04,
        7.44563703e+03, -2.02831115e+02,  2.32271366e+03,  1.20458921e+04,
        6.10775586e+03,  2.73076495e+03,  1.17619702e+04,  8.33509889e+03,
        7.31998467e+03,  5.09479776e+03,  2.83362371e+03,  3.19778042e+04,
        3.22979849e+03,  9.06310058e+03,  4.70091757e+03,  1.24845886e+04,
        1.49653272e+04,  7.43900388e+03,  2.69137528e+04,  1.42890886e+04,
        1.74534894e+04,  1.19956737e+04,  1.56285848e+03,  8.47115098e+03,
        1.89789941e+04,  7.75757849e+03,  6.24509839e+03,  2.32263149e+03,
        2.64227213e+04,  1.44739879e+04,  1.05710942e+04,  1.62834582e+04,
        1.21067301e+04,  1.14271890e+04,  3.95113022e+04,  6.24457881e+03,
        1.17610108e+04,  3.15572075e+03,  1.51834988e+04,  4.57738288e+03,
        5.10208823e+03,  5.33609478e+03,  1.53840075e+04,  8.94770798e+03,
        8.52222219e+03,  3.41622874e+04,  6.78287771e+03,  2.65443292e+04,
        2.69337275e+04,  1.54681923e+04,  8.69711159e+03,  3.40099601e+04,
        1.02860070e+04,  3.73565854e+03,  1.55799060e+04,  2.97896387e+04,
        2.95544539e+04,  2.81355957e+04,  9.61789556e+03,  3.24495021e+04,
        4.50321038e+03,  2.54475681e+04,  5.36031177e+03,  2.82108122e+04,
        7.23802872e+03,  1.47033982e+04])
```

11. Finding the R2 Value

(input)/ (Output)

```
r_score
```

```
0.6188543215916411
```

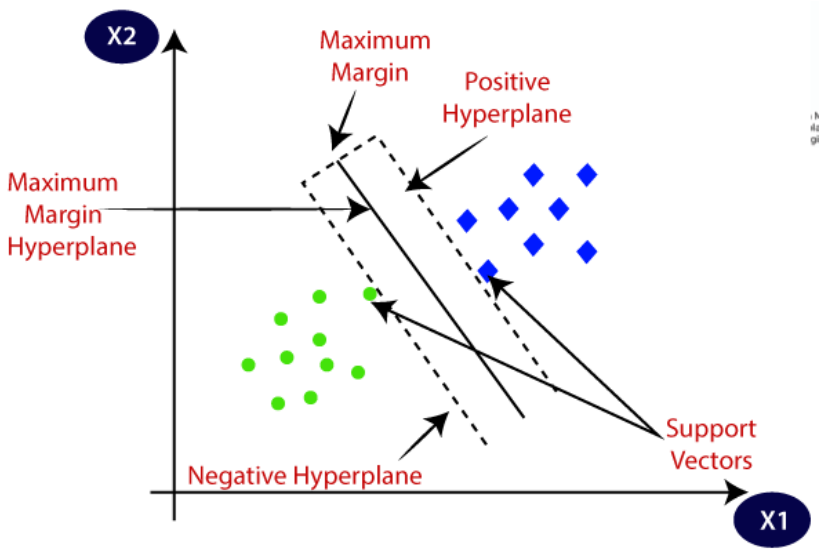
A) Conclusion for Multiple Linear Regression

The **Multiple Linear Regression** use R^2value **-0.61**

Support Vector Regression

B)Support Vector Regression-The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

MATPLOT-(**Support Vector Regression)**



1. Finding the R2 Value in Standard Scaler - import SVR (input)/ (Output)

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.svm import SVR
regressor = SVR(kernel ='rbf',C=3000)
regressor.fit(X_train, y_train)
```

```
SVR(C=3000)
```

2.Finding the R2 Value in Using Grid

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVR
param_grid = {'kernel':['rbf','poly','sigmoid','linear'],
              'C':[10,100,1000,2000,3000],'gamma':['auto','scale']}



grid = GridSearchCV(SVR(), param_grid, refit = True, verbose = 3,n_jobs=-1)

# fitting the model for grid search
grid.fit(X_train, y_train)
```

B) Conclusion for Support Vector Regression

```
# print best parameter after tuning
#print(grid.best_params_)
re=grid.cv_results_
#print(re)
grid_predictions = grid.predict(X_test)

# print classification report
from sklearn.metrics import r2_score
r_score=r2_score(y_test,grid_predictions)

print("The R_score value for best parameter {}:".format(grid.best_params_),r_score)
```

```
The R_score value for best parameter {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}: 0.8553472249274324
```

The **SVM Regression** use R^2value (nonlinear (Rbf) and hyper parameter (C3000)) **=0.85**

Decision Tree Regression

C) **Decision Tree Regression-**Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application. It is a tree-structured classifier with three types of nodes.

MATPLOT-(**Decision Tree Regression)**

1.Finding the R2 Value in Standard Scaler- import Decision Tree Regressor

(input)/ (Output)

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(criterion="mae",max_features="sqrt",random_state = 0)
regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(criterion='mae', max_features='sqrt', random_state=0)
```

2.Finding the R2 Value in Using Grid

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
param_grid = {'criterion':['mse','mae','friedman_mse'],
              'max_features': ['auto','sqrt','log2'],
              'splitter':['best','random']}


grid = GridSearchCV(DecisionTreeRegressor(), param_grid, refit = True, verbose = 3,n_jobs=-1)

# fitting the model for grid search
grid.fit(X_train, y_train)
```

C)Conclusion for Decision Tree Regression

```
# print best parameter after tuning
#print(grid.best_params_)
re=grid.cv_results_
#print(re)
grid_predictions = grid.predict(X_test)

# print classification report
from sklearn.metrics import r2_score
r_score=r2_score(y_test,grid_predictions)

print("The R_score value for best parameter {}:".format(grid.best_params_),r_score)
```

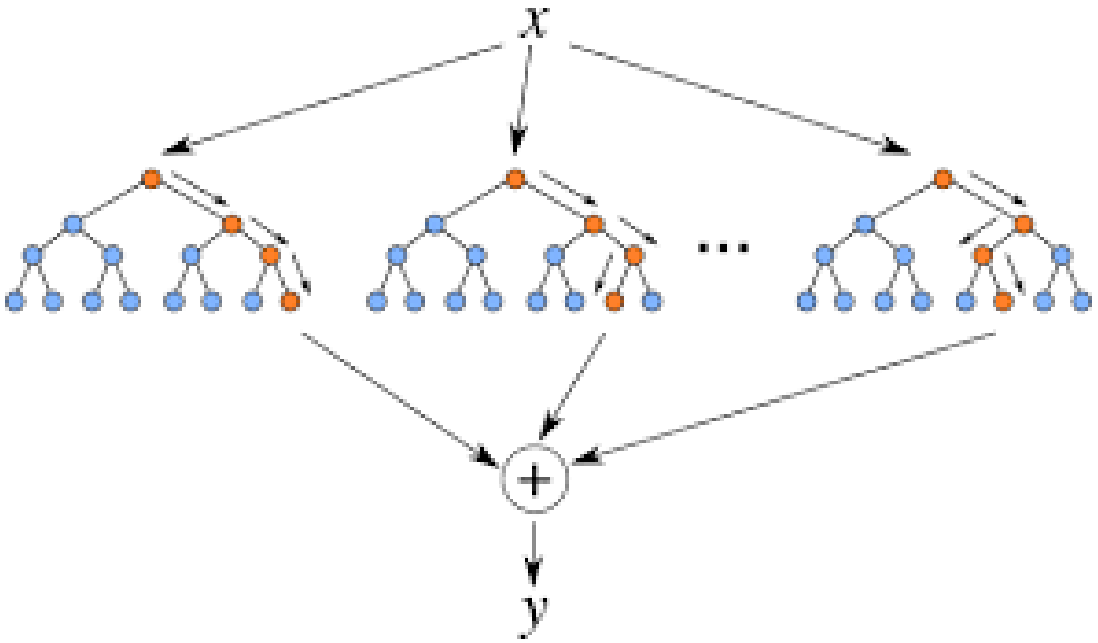
```
The R_score value for best parameter {'criterion': 'friedman_mse', 'max_features': 'auto', 'splitter': 'best'}: 0.8096986725661
193
```

The **Decision Tree** Regression use R^2value (Mae, auto, random) =**0.809**

Random Forest Regression

D) **Random Forest Regression**-Random forest is a type of supervised learning algorithm that uses ensemble methods (bagging) to solve both regression and classification problems. The algorithm operates by constructing a multitude of decision trees at training time and outputting the mean/mode of prediction of the individual trees.

MATPLOT-(Random Forest Regression)



1.Finding the R2 Value in Standard Scaler- import Random Forest Regression

(input)/ (Output)

```
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 = 10, random_state = 0)
regressor.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=10, random_state=0)
```

2.Finding the R2 Value in Using List

```
from sklearn.model_selection import GridSearchCV
#from sklearn.tree import DecisionTreeRegressor
param_grid = {'criterion':['mse','mae'],
              'max_features': ['auto','sqrt','log2'],
              'n_estimators':[10,100]}


grid = GridSearchCV(RandomForestRegressor(), param_grid, refit = True, verbose = 3,n_jobs=-1)

# fitting the model for grid search
grid.fit(X_train, y_train)
```

D)Conclusion for Decision Random Forest

```
# print best parameter after tuning
#print(grid.best_params_)
re=grid.cv_results_
#print(re)
grid_predictions = grid.predict(X_test)

# print classification report
from sklearn.metrics import r2_score
r_score=r2_score(y_test,grid_predictions)

print("The R_score value for best parameter {}:".format(grid.best_params_),r_score)
```

```
The R_score value for best parameter {'criterion': 'mse', 'max_features': 'auto', 'n_estimators': 100}: 0.8950833469401109
```

The **Random Forest** Regression use R^2value (Mae, auto, random) =**0.895**

8. FINAL MODEL

Using regression algorithms, to calculate the r squared value to know the best performance of a model.

ALGORITHM	R2 VALUE
Multiple Linear Regression	0.61
SVM	0.85
Decision Tree	0.806
Random Forest	0.89

Tabulation of R_2 value using Regression Algorithms

The final machine learning best method of Regression:

1. **Random Forest** R^2value (Criterion,mse,max features,auto,n_estimators 100) =**0.895**

Thank You

M.Ezhilarasan form hope