Introduction

Cement concrete is the most important material in civil engineering. The quality of cement concrete is determined by its compressive strength, which is measured using a conventional crushing test on a concrete cube sample using a Universal testing machine. Standardized, popular, and straightforward means of measurement, the compressive strength of concrete at 28 days offers a convenient metric of engineering performance that forms a key input in structural design and quality control.

Although concrete's strength is governed largely by the water-cement ratio (w/c), it is also affected by other features, such as chemical and mineral admixtures, cement type and quantity, aggregates types and quantity, and entrained air.

Altogether, the high number of features influencing concrete's strength and the fact that the effects of individual features may be nonlinear, competitive, and/or non-additive make reliable prediction of strength development in concrete extremely challenging. Despite decades of research, no robust, accurate models that can precisely, accurately, and reliably predict concrete's strength are currently available.

Objective

The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate. Machine learning (ML) offers an attractive option to develop data-driven models by "learning from example" based on existing data sets to predict the compressive strength of cement concrete.

Dataset

We imported dataset from website Kaggle.com. The link for the same is as follows

https://www.kaggle.com/datasets/elikplim/concrete-compressive-strength-data-set

This dataset includes these 9 features along with the names they are renamed with–

Cement (component 1) (kg in a m^3 mixture) -cement

Blast Furnace Slag (component 2) (kg in a m^3 mixture)-slag

Fly Ash (component 3) (kg in a m^3 mixture)- Fly ash

Water (component 4) (kg in a m^3 mixture)- Water

Superplasticizer (component 5) (kg in a m^3 mixture)-SP

Coarse Aggregate (component 6) (kg in a m^3 mixture)-CA

Fine Aggregate (component 7) (kg in a m^3 mixture)-FA

Age (day)-age

Concrete compressive strength (MPa, megapascals)- Strength

```
df.describe()
```

	cement	slag	fly ash	water	SP	CA	FA	age	strength
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.580485	45.662136	35.817961
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.175980	63.169912	16.705742
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.000000	56.000000	46.135000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	365.000000	82.600000

Libraries

We have used following libraries for the various purposes

Numpy

Pandas

Scipy

Matplotlib,

Seaborn,

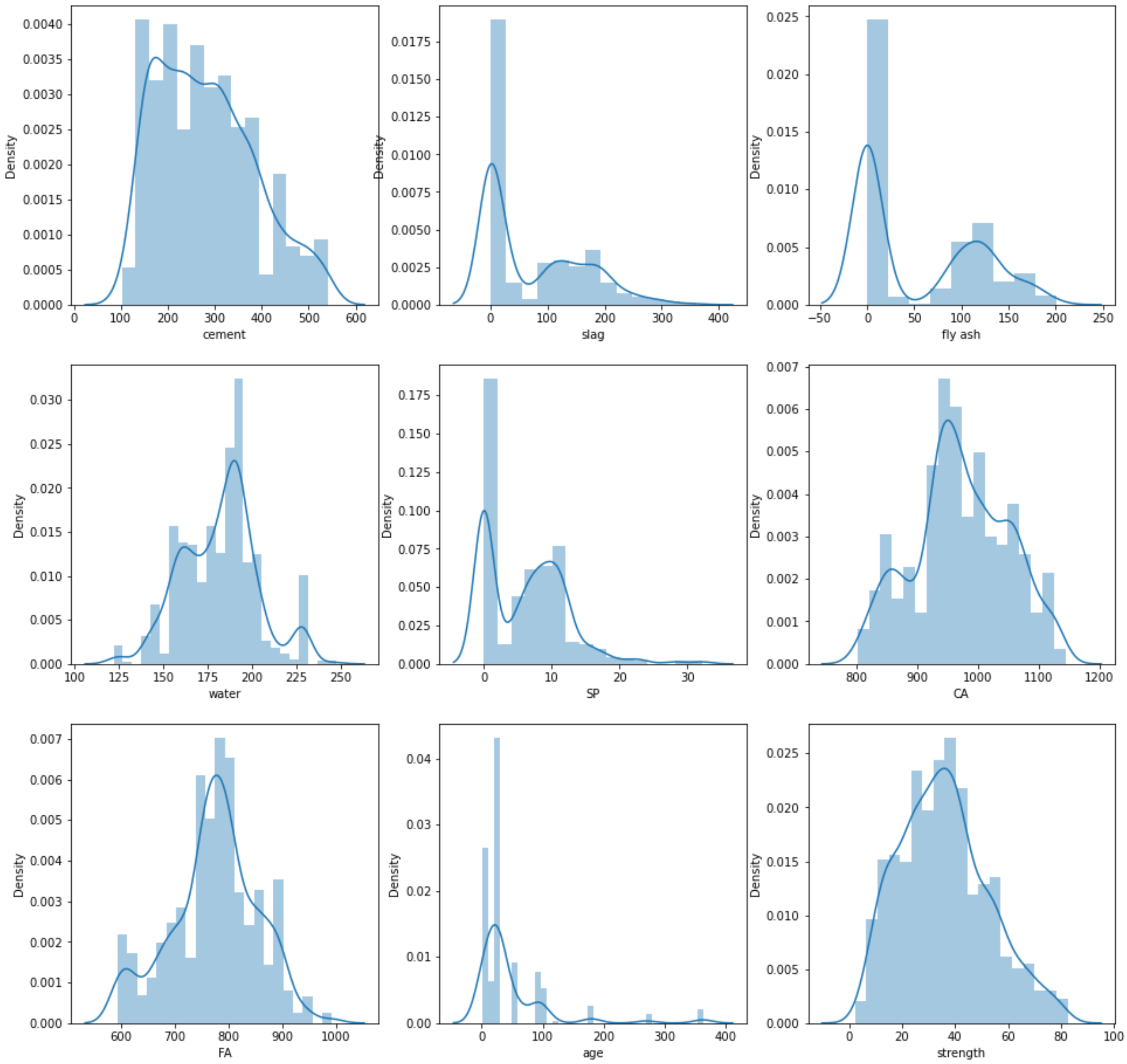
Sklearn

Tools

We have used Jupyter Notebook for coding.

Data visualization

```
fig,allDistPlots=plt.subplots(3,3,figsize=(16,16))
i=0
j=0
for feature in df.columns:
    sbn.distplot(df[feature],ax=allDistPlots[i][j])
    j+=1
    if(j>2):
        i+=1
        j=0
```



Feature engineering

Checking Null values in dataset

We have used following command to check the Null values

```
df[df.isnull().any(axis=1)]
```

cement	slag	fly ash	water	SP	CA	FA	age	strength

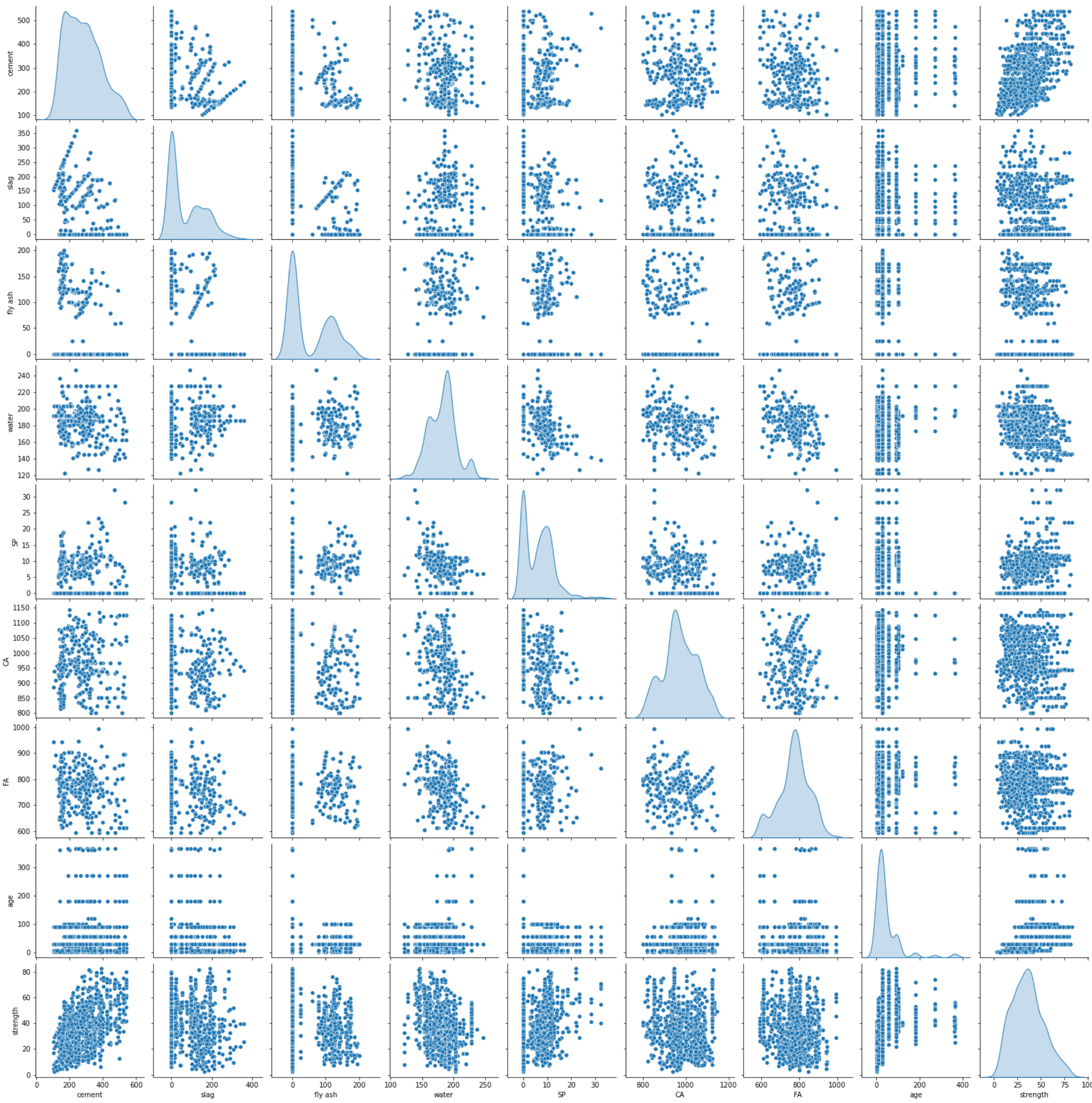
We found that in our dataset there was no Null values.

checking correlation between independent and dependent feature

We know that if two features are correlated with each other about say more than or equal to 80% then one of them can be dropped. And in this way we can reduce the complexity of the machine learning algorithms as well as computation.

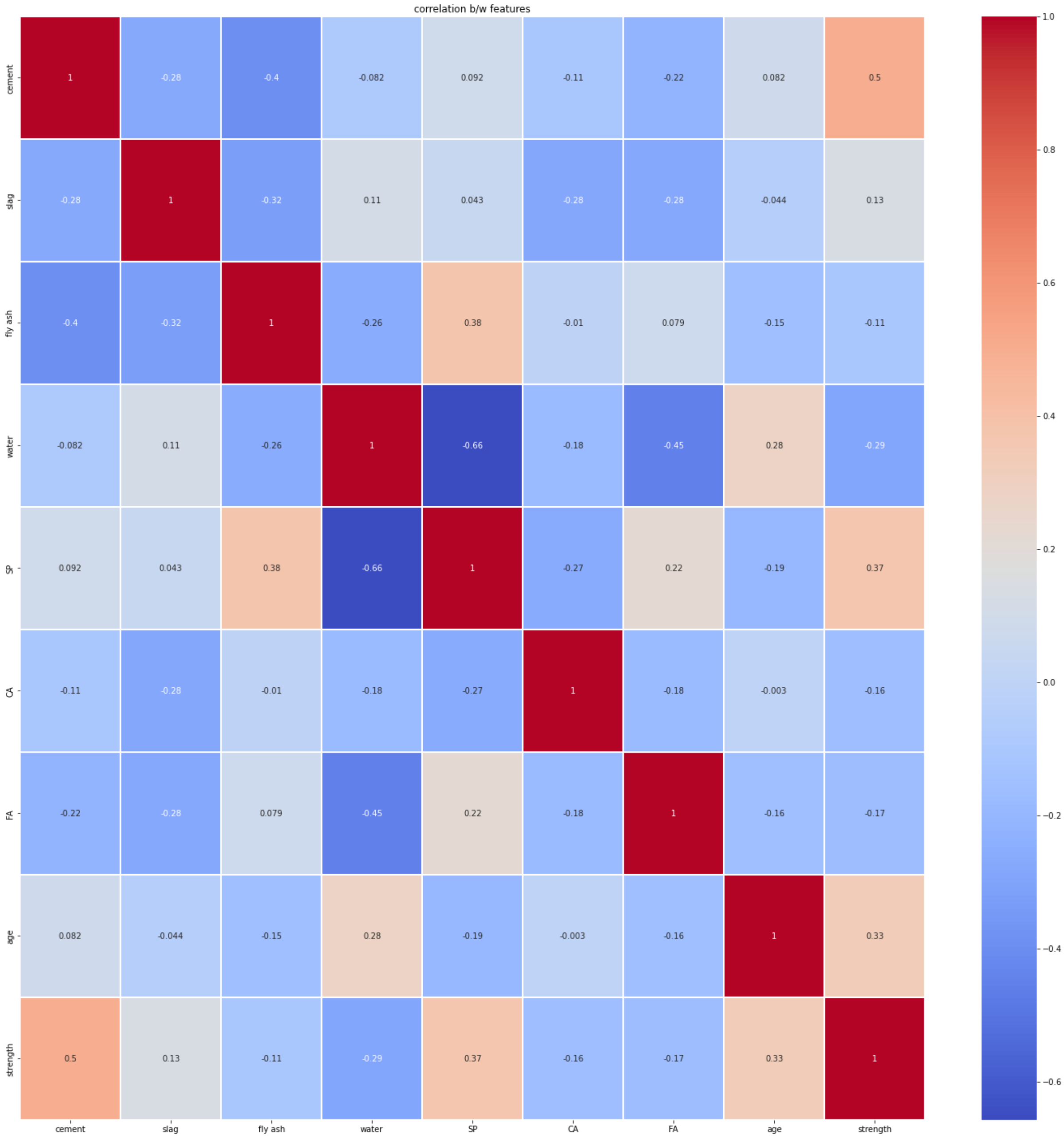
So for checking correlation between the features we have used seaborn library. From the seaborn library first we used the pairplot and then heatmap.

From the pairplot we came to know about the existing relationship between different features.



Here we can see there exist no definite pattern between the features. Now lets check the correlation using the heatmap

```python
plt.figure(figsize=(25,25))
sbn.heatmap(df.corr(),cmap='coolwarm',annot=True,linewidth=2)
plt.title('correlation b/w features')
```



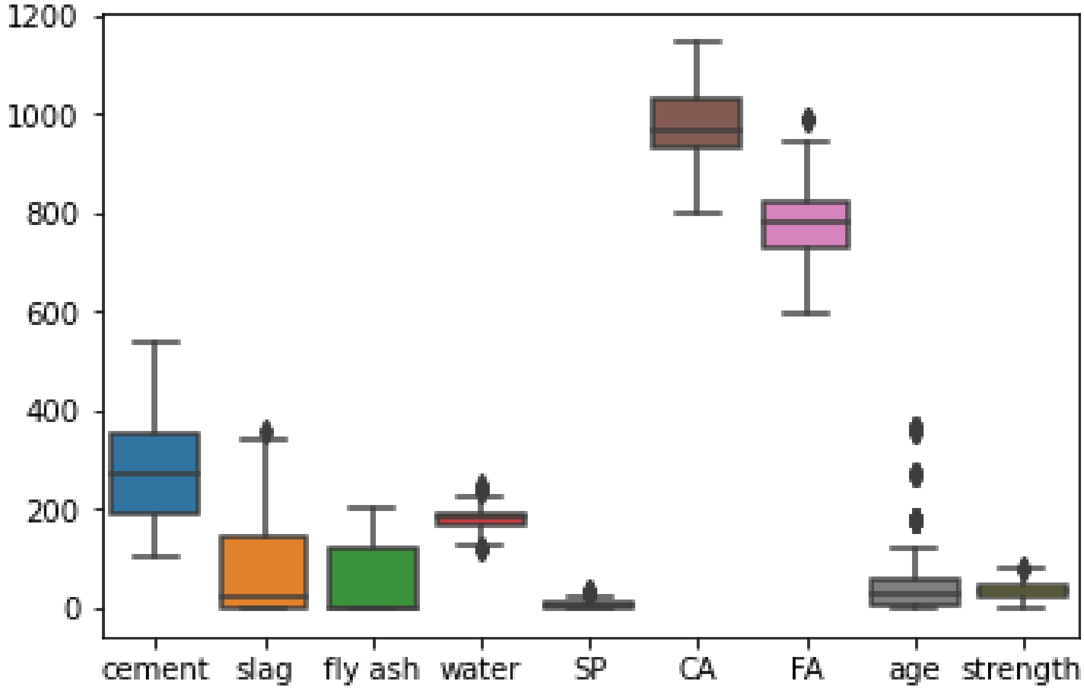
Maximum correlation between independent and dependent feature is 50% and that is between cement and strength.

And maximum correlation between independent features super plasticizer and fly ash is 38%.

So in this case as our correction between independent features is not enough therefore we will not drop any of the independent feature.

checking outliers in data

This is the boxplot for the dataset



For declaring the data point as a outlier we have considered 3 standard deviation approach i.e. our data point will be outlier if it lies $>3\sigma$ and $<-3\sigma$

So in our dataset we have outliers as follows

```
Outliers in  cement  are:  0
Outliers in  slag  are:  4
Outliers in  fly ash  are:  0
Outliers in  water  are:  2
Outliers in  SP  are:  10
Outliers in  CA  are:  0
Outliers in  FA  are:  0
Outliers in  age  are:  33
Outliers in  strength  are:  0
```

Here we are not considering outliers in "age" since compression test on concrete block can be done after enough many days.

So after removing the outliers our dataset has reduced to

```
dfo.describe()
```

	cement	slag	fly ash	water	SP	CA	FA	age	strength
count	1014.000000	1014.000000	1014.000000	1014.000000	1014.000000	1014.000000	1014.000000	1014.000000	1014.000000
mean	279.282347	72.919132	54.902860	181.833925	5.992899	974.442604	773.224162	45.893491	35.645128
std	102.930632	84.976165	64.176615	20.913290	5.507277	77.225729	79.881805	63.543679	16.666211
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	191.025000	0.000000	0.000000	164.900000	0.000000	932.000000	733.250000	7.000000	23.550000
50%	272.800000	22.000000	0.000000	185.000000	6.400000	968.000000	779.300000	28.000000	34.080000
75%	349.750000	142.950000	118.300000	192.000000	10.100000	1030.000000	822.150000	56.000000	45.807500
max	540.000000	316.100000	200.100000	237.000000	23.400000	1145.000000	992.600000	365.000000	82.600000

Feature scaling

We have used StandardScaler for feature scaling

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
print(x)
```

Training and testing of models

To avoid any risk of overfitting, a fraction of the data points (randomly chosen) are hidden from the models and are used as a “test set” to assess the accuracy of each model, that is to minimize variance and bias. The test set is formed by randomly selecting 20% of the data points within the data set. The rest (that is, 80%) are used as a “training set” that is, to train “by example” the ML models.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=1)
```

Building different Models

1. Multiple linear regression

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
model_lr = LinearRegression()
model_lr.fit(x_train, y_train)
```

```
model_lr.coef_
```

```
array([12.62113679,  9.01532994,  5.48934544, -3.52539426,  2.06315885,
        1.40045011,  1.32206482,  8.01857701])
```

```
model_lr.intercept_
```

```
35.99638493960809
```

Mean squared error = 116.23175624089491
Accuracy = 64.061273

2. Random Forest

```
from sklearn.ensemble import RandomForestRegressor
```

```
model_rf=RandomForestRegressor()
```

```
model_rf.fit(x_train,y_train)
```

Mean squared error = 16.850752823753872
Accuracy = 89.388453

3. KFold cross validation on random forest

```
from sklearn.model_selection import KFold
```

```
k = 50

kfold = KFold(n_splits=k, shuffle=True, random_state=1)
res_kfold_rf = cross_val_score(model_rf, x, y, cv=kfold)
res_kfold_rf
```

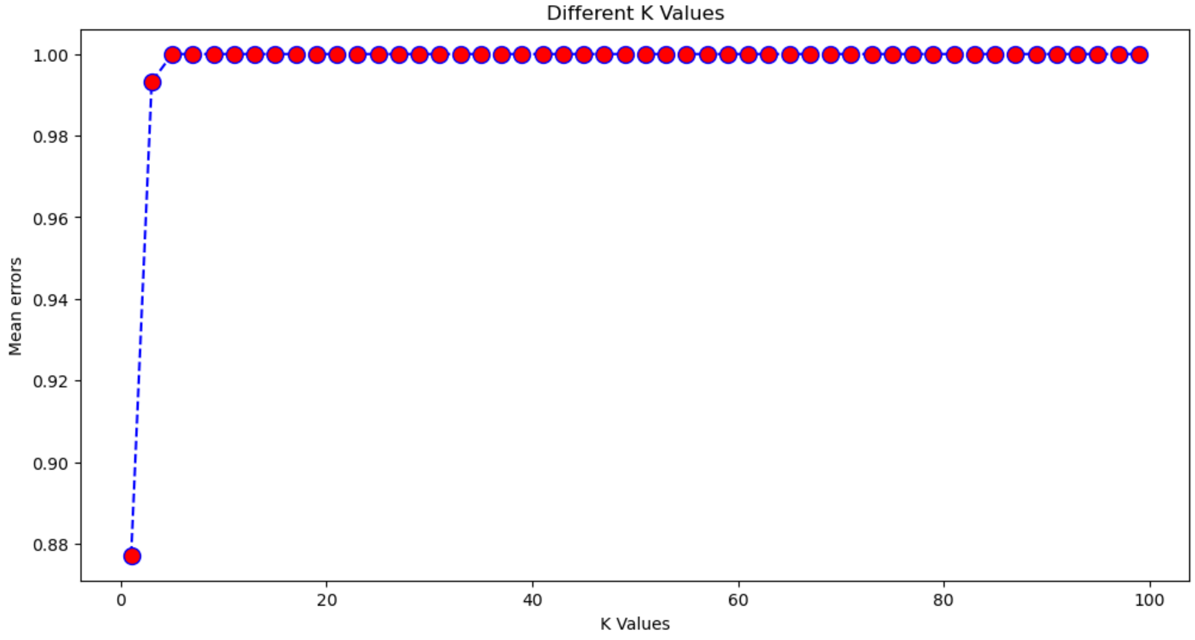
Accuracy= 91.072524

4. KNN Regressor

```
from sklearn.neighbors import KNeighborsRegressor

diff_k=[]
for i in range(1,41):
  knn = KNeighborsRegressor(n_neighbors=i)
  knn.fit(x_train, y_train)
  pred_i = knn.predict(x_test)
  diff_k.append(np.mean(pred_i != y_test))
```

To find optimum k value we used elbow method



We selected k=3 from above graph

```
model_knn = KNeighborsRegressor(n_neighbors=3)
model_knn.fit(x_train, y_train)
```

Mean squared error = 64.27853056157635
Accuracy = 74.127321

5. Support Vector Machine

```
from sklearn.svm import SVR
model_svm = SVR(kernel='rbf')
model_svm.fit(x_train, y_train)
y_pred_svm = model_svm.predict(x_test)
model_svm.score(x_train, y_train)
```

Mean squared error = 82.36749682316874
Accuracy = 70.933823

6. KFold Support Vector Machine

```
k=20
kfold = KFold(n_splits=20,shuffle=True,random_state=3)
res_kfold_svm = cross_val_score(model_svm, x, y, cv=kfold)
res_kfold_svm
```

Accuracy = 22.511145

7. Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor
model_dec=DecisionTreeRegressor()
model_dec.fit(x_train,y_train)
```

```
Feature importance:
          Importance
cement      0.352388
slag        0.073357
fly ash     0.008321
water       0.071057
SP          0.091220
CA          0.032955
FA          0.035638
age         0.335063
```

Accuracy = 75.82200129899374

8. Pruning Decision Tree

```
from scipy import stats
Xscaled=stats.zscore(X)
Xscaled_df=pd.DataFrame(Xscaled,columns=dataset.columns)
```

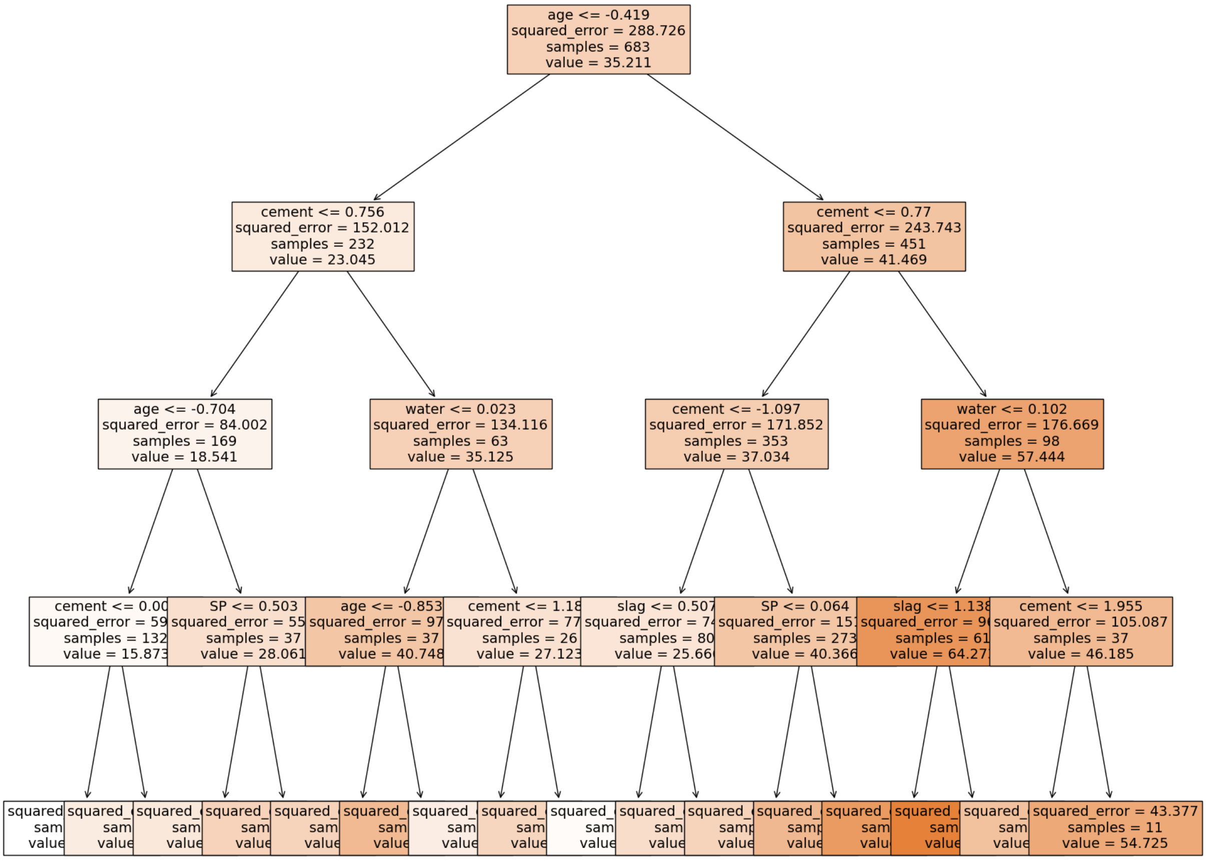
```
x_train,x_test,y_train,y_test=train_test_split(Xscaled,Y,test_size=0.3,random_state=5)
```

```
dt_prun_model=tree.DecisionTreeRegressor(max_depth=4,random_state=1,min_samples_leaf=5)
dt_prun_model.fit(x_train,y_train)
```

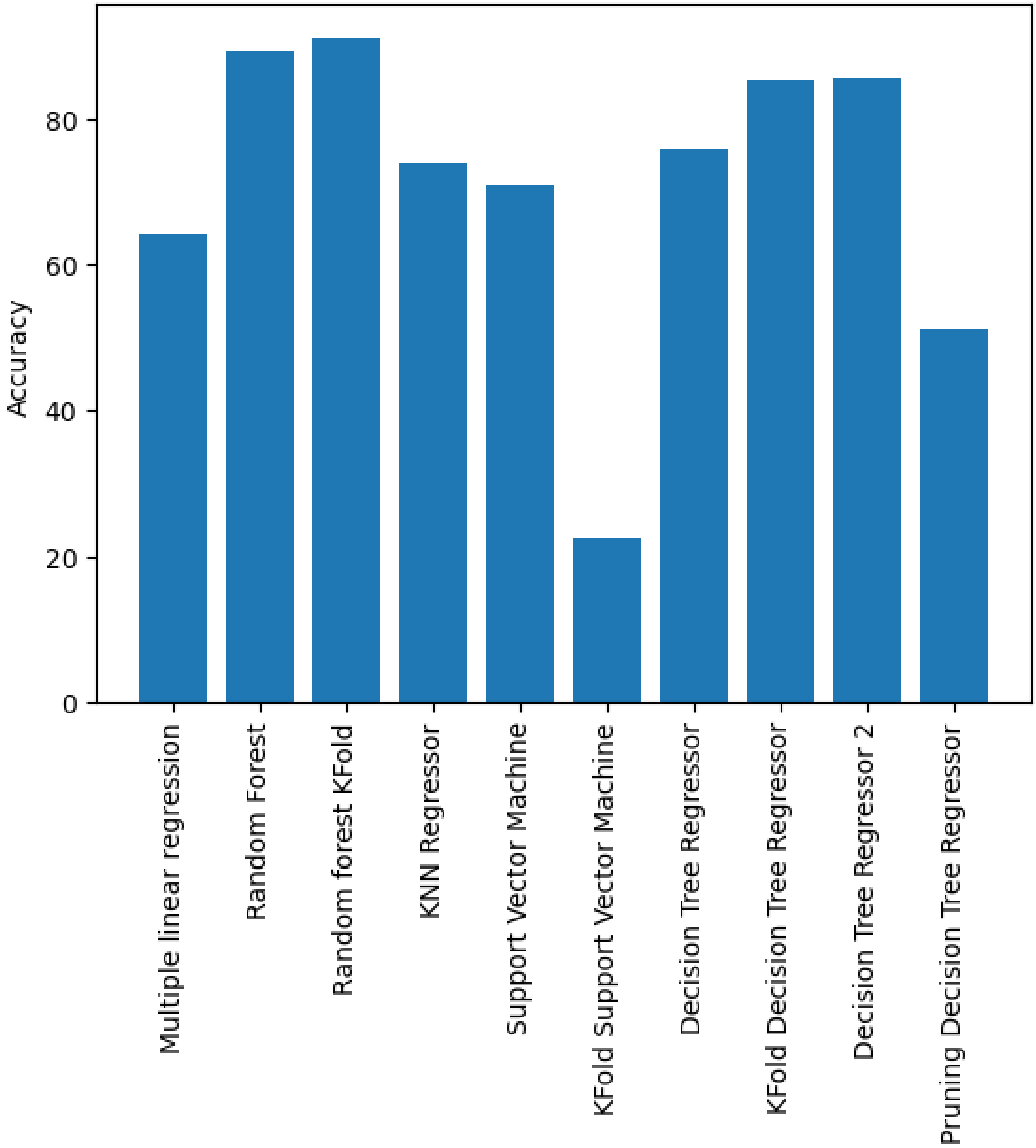
```
Feature importance:
          Importance
cement      0.421864
slag        0.031175
fly ash     0.000000
water       0.070190
SP          0.086034
CA          0.000000
FA          0.000000
age         0.390738
```

Accuracy = 51.242091785808505

Plotting Decision Tree



Accuracy Comparison of all algorithms



So in this model we conclude that Random forest k-fold gives max accuracy