

# Concrete Compressive Strength Prediction

---

BY Sanket Jagtap

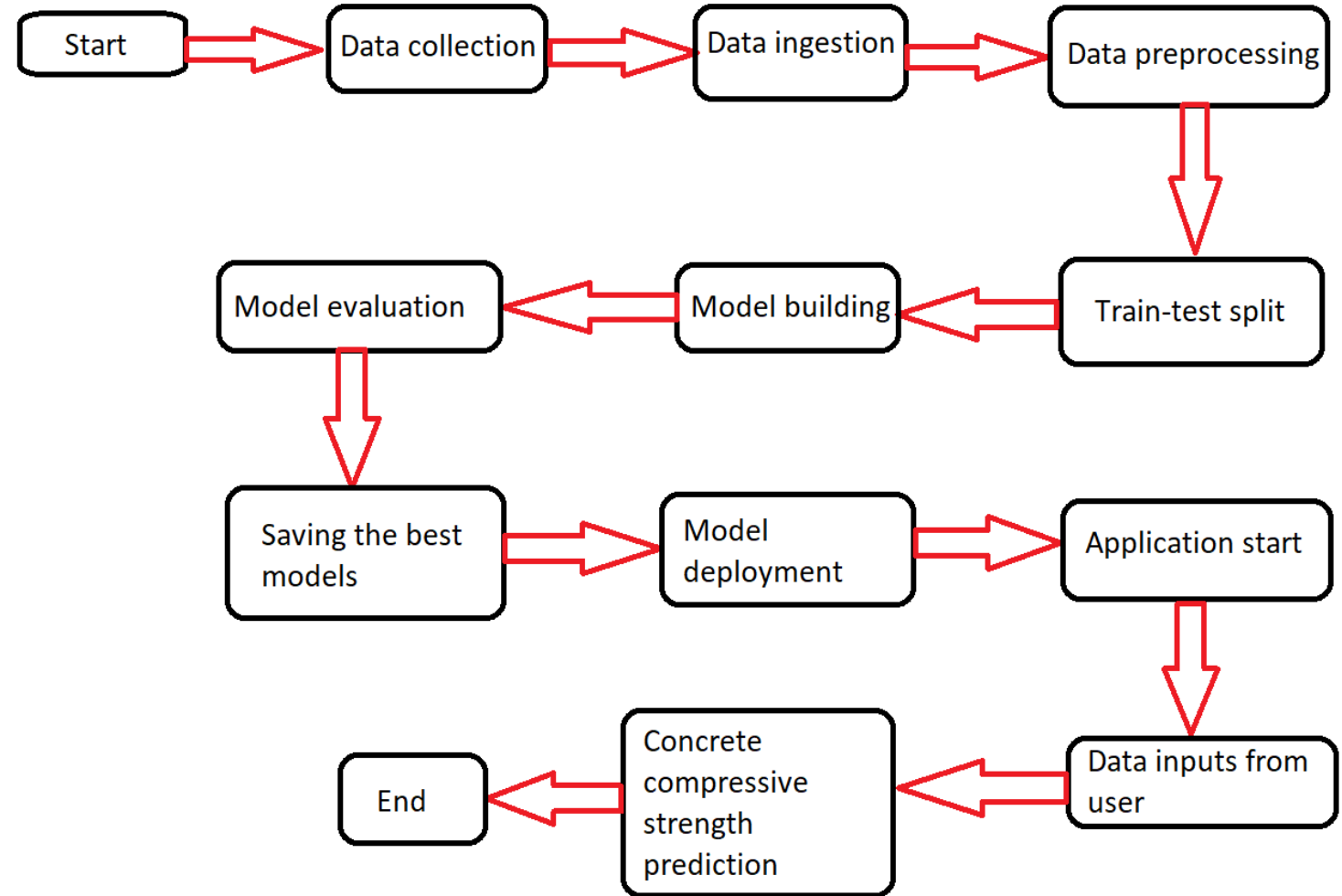
# Objective

---



Being one of the most frequently used building materials, the quality of concrete is determined by its compressive strength, which is measured by crushing a concrete cube or a cylinder until it starts cracking and crushed. The pressure at which the concrete cube or a cylinder starts cracking and eventually crushes is called the Concrete compressive strength and is measured in megapascals (MPa). It takes a long period of 28 days to test like this. With the help of Data science and the Machine learning technology, I developed an application, which allows an engineer to determine the strength of a concrete in just a few seconds of time.

# Architecture



# Data set

---

For training and testing the model, we used the public data set available in Kaggle, “Concrete Compressive Strength Data Set” by Ahiale Darlington.

URL -  
<https://www.kaggle.com/elikplim/concrete-compressive-strength-data-set>

Name	Data Type	Measurement	Description
Cement	Quantitative	kg in a m3 mixture	Input variable
Blast Furnace Slag	Quantitative	kg in a m3 mixture	Input variable
Fly Ash	Quantitative	kg in a m3 mixture	Input variable
Water	Quantitative	kg in a m3 mixture	Input variable
Superplasticizer	Quantitative	kg in a m3 mixture	Input variable
Coarse Aggregate	Quantitative	kg in a m3 mixture	Input variable
Fine Aggregate	Quantitative	kg in a m3 mixture	Input variable
Age	Quantitative	Days (1~365)	Input variable
Concrete Compressive Strength	Quantitative	megapascals (MPa)	Output variable

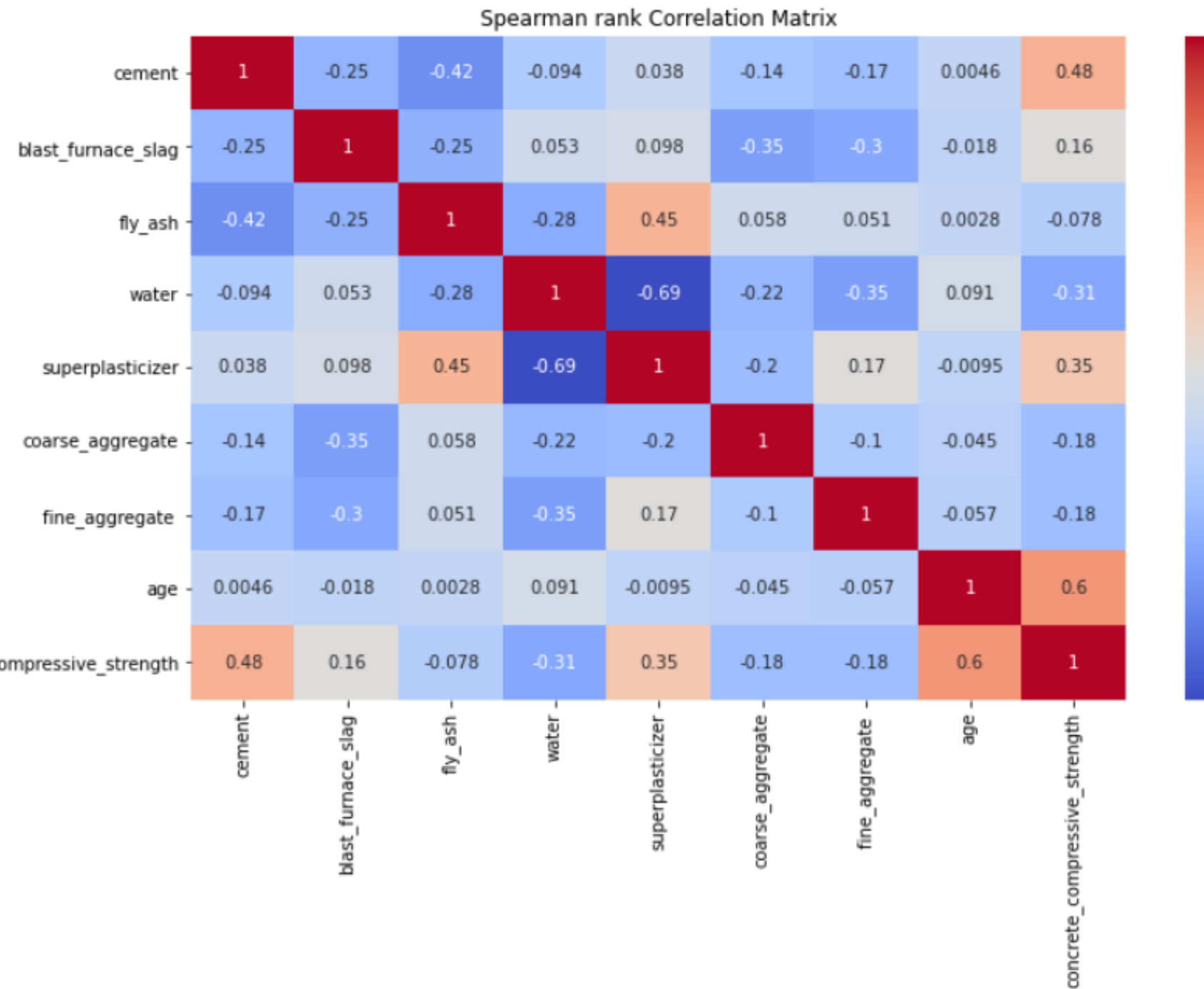
# EDA – Data cleaning

---

- The given data set has 9 features and each one is quantitative in nature.
- This data set doesn't have any missing values.
- The shape of the data set is 1030 rows × 9 columns.

```
In [4]: #info of the dataframe  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1030 entries, 0 to 1029  
Data columns (total 9 columns):  
#   Column                                Non-Null Count  Dtype  
---  -  
0   cement                               1030 non-null   float64  
1   blast_furnace_slag                   1030 non-null   float64  
2   fly_ash                              1030 non-null   float64  
3   water                                1030 non-null   float64  
4   superplasticizer                     1030 non-null   float64  
5   coarse_aggregate                     1030 non-null   float64  
6   fine_aggregate                       1030 non-null   float64  
7   age                                   1030 non-null   int64  
8   concrete_compressive_strength         1030 non-null   float64  
dtypes: float64(8), int64(1)  
memory usage: 72.5 KB
```



## EDA – Data visualization

- Age, Cement, super plasticizer and Blast furnace slag are in positive correlation to the Concrete compressive strength.
- While water, coarse aggregate and fine aggregate are in negative correlation to the Concrete compressive strength.
- Super plasticizer is having negative correlation to the water.

# EDA – Outlier treatment for “Age” column

- ‘Age’ is the only column having more number of outliers. I used Interquartile range (IQR) method to treat the outliers.
- Here, first I considered the values within the upper and lower extremes of the ‘age’ column from the main data-frame ‘df’, inclusive of those extreme values and stored them in a data-frame called “data\_inc”.
- Then, again I considered the values of ‘age’ column within the upper and lower from the main data-frame ‘df’, but this time excluding those extreme values and stored them in a data-frame called “data\_esc”.
- Then, concatenated both the data-frames into a single data-frame. The benefits of doing this are:
  - Outliers removed
  - The no.of rows in the dataframe increased to 1942 (it’s like copying within the dataset), which helps the models to avoid overfitting.

```
In [14]: #using iqr method, considering both including and excluding the lower and upper limits into two separate dataframes,  
# then merging both the dataframes into one.  
# This increases the data size so that a Machine Learning model can be trained efficiently.  
q1 = df['age'].quantile(0.25)  
q2 = df['age'].quantile(0.50)  
q3 = df['age'].quantile(0.75)  
IQR = q3-q1  
  
lower_limit = q1-1.5*IQR  
upper_limit = q3+1.5*IQR  
  
data_inc = df.loc[(df['age']>=lower_limit) & (df['age']<=upper_limit)]  
data_esc = df.loc[(df['age']>lower_limit) & (df['age']<upper_limit)]  
df = pd.concat([data_inc,data_esc])
```

# EDA – Outlier treatment for “Age” column - code



# Data pre-processing

- Split the data-frame into the training (70% of the data) and testing (30% of the data) data-frames respectively.
- For building linear regression models, used the scaled data obtained by scaling the features using the Standard scaler. Tree models use the original data i.e., without scaling as they aren't affected by the feature scaling.
- Then both the training and testing data-frames were further split into  $X_{train}$ ,  $X_{test}$ ,  $y_{train}$  and  $y_{test}$ . Here the data-frames with  $X$  indicate independent features while those with  $y$  indicate the dependent or the target feature.

# Model building and evaluation

---

Experimented with Linear regression models and the tree models, got the following results. XGBoost regressor is the best model followed by the RandomForest regressor model.

Algorithm	Train_R2 score	Train_Adj_R2 score	Train_RMSE score	Test_R2 score	Test_Adj_R2 score	Test_RMSE score
Linear Regression_BE	0.750781264	0.749675264	0.499218125	0.773336421	0.770975342	0.486707531
Linear Regression_RFE	0.750834519	0.749357982	0.499164784	0.773392486	0.770234193	0.486647333
Linear Regression_Lasso	0.750808903	0.749332215	0.499190441	0.773235676	0.770075198	0.486815682
Decision tree regressor	0.9631484	0.963039532	3.216372805	0.917452711	0.916881449	4.921152341
Decision tree regressor_post pruning	0.863353273	0.86294959	6.193523976	0.844822215	0.84374832	6.747302403
Random Forest regressor	0.961607367	0.961465488	3.282934001	0.941797447	0.941293092	4.132248383
Adaboost regressor	0.833317523	0.832453883	6.840426466	0.831264515	0.829210344	7.03588306
Gradient Boost regressor	0.921889029	0.921658273	4.682677224	0.908404644	0.907770766	5.183846434
XGBoost regressor	0.943596049	0.943345736	3.979174344	0.943253834	0.942662729	4.080220759

```
The best estimator for the XGBoost regressor is XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
    colsample_bynode=1, colsample_bytree=0.5, gamma=0.2, gpu_id=-1,
    importance_type='gain', interaction_constraints='',
    learning_rate=1, max_delta_step=0, max_depth=4, min_child_weight=4,
    missing=nan, monotone_constraints='()', n_estimators=100,
    n_jobs=16, num_parallel_tree=1, random_state=0, reg_alpha=0.5,
    reg_lambda=1000, scale_pos_weight=1, subsample=0.9,
    tree_method='exact', validate_parameters=1, verbosity=None)
```

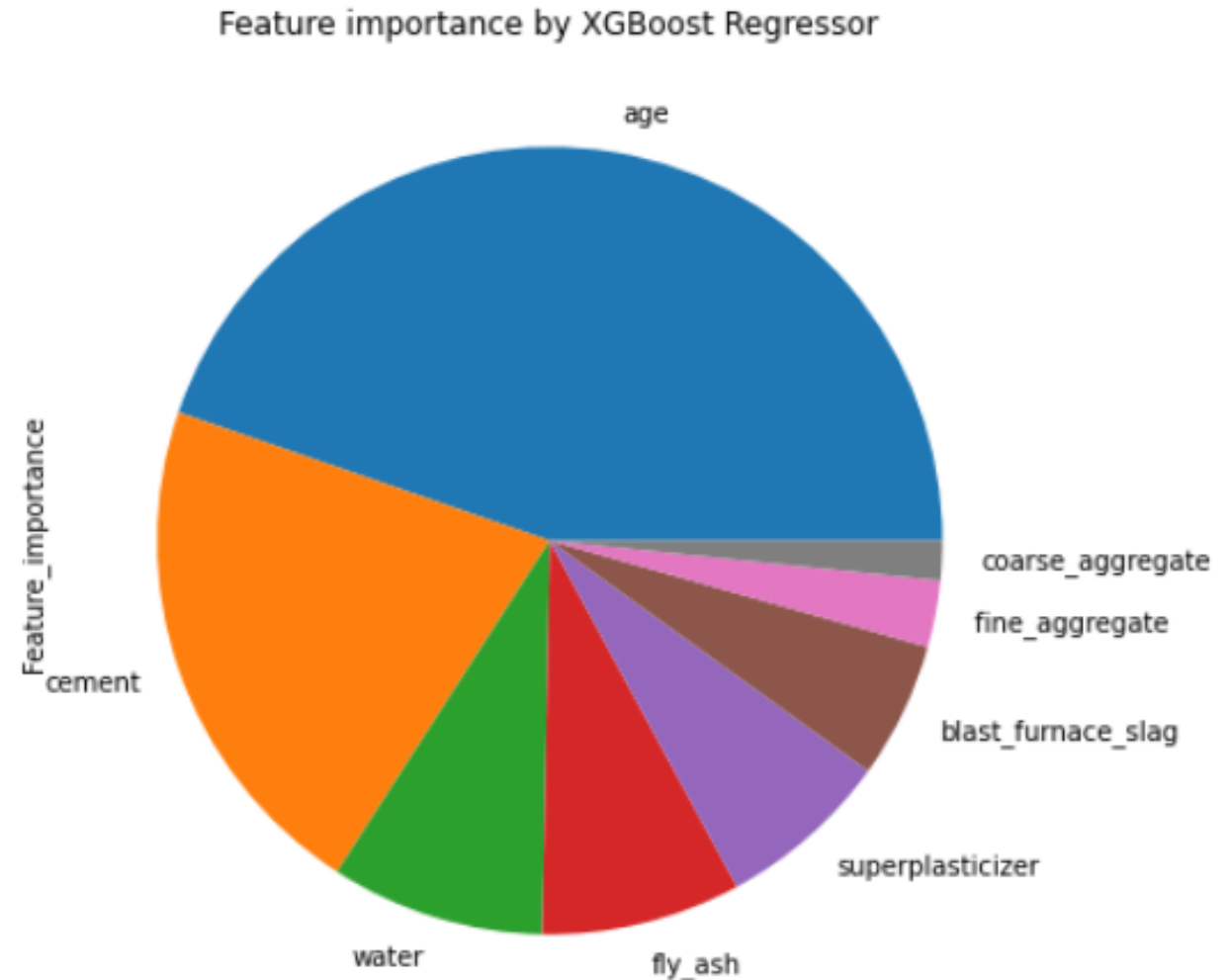
# XGBoost regressor's parameters

---

## Feature importance by the XGBoost regressor

---

Since, both the “fine\_aggregate” and the “coarse\_aggregate” are contributing less to the prediction of the Concrete compressive strength, I dropped them and considered only the top 6 features






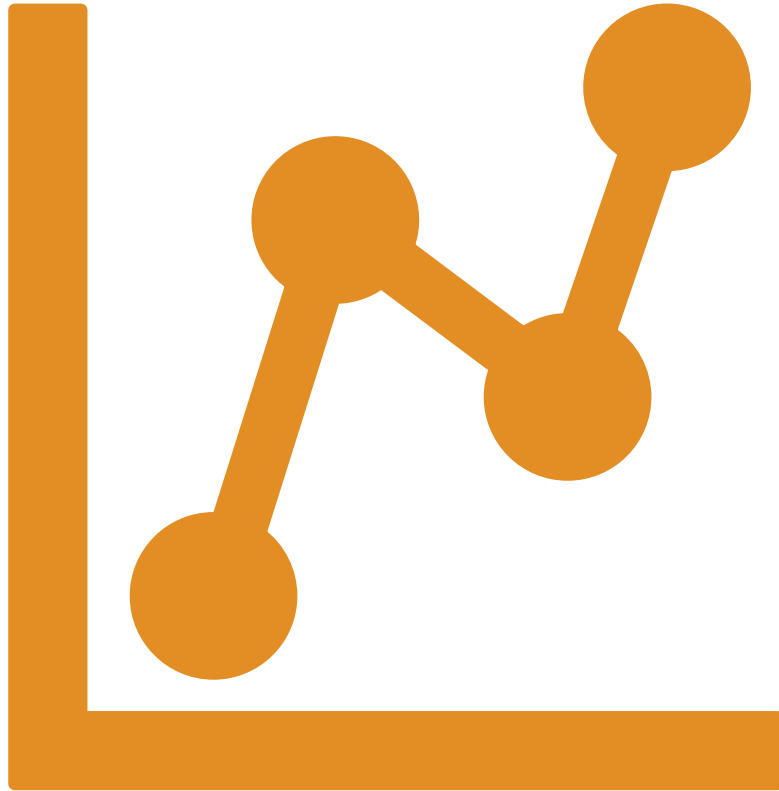
# Model deployment

---

- Saved XGBoost regressor model and the RandomForest regressor model into the “models” directory. Then deployed the XGBoost regressor model using the Flask and linked to web application which was designed using HTML 5.
- Deployed on web using GitHub, Unicorn and Koyeb.
- This application can be accessed using the URL:- <https://eventual-elnore-sanketjagtap-7e11777b.koyeb.app/>

### Concrete Compressive Strength Prediction

 LINKEDIN  KAGGLE  GITHUB



# Key performance indicators (KPI)

---

- Time and workload reduction using the regression models.
- Comparison of the  $R^2$  scores and the Adjusted  $R^2$  scores of the model on both the training and the testing data.
- Comparison of the RMSE scores of the model on both the training and the testing data.
- Low water to cement ratio is proportional to the strength of the concrete.

# Q & A

1) What's the source of data?

Answer: The data for training the model is taken from Kaggle - <https://www.kaggle.com/elikplim/concrete-compressive-strength-data-set>.

2) What's the complete flow you followed in this project?

Answer: Please refer slide 3 for the better understanding.

3) What techniques were you using for data-preprocessing?

Answer: 1. Visualizing the relation ships between the target variable and each predictor variable.

2. Visualizing the correlation among all the variables.

3. Removing outliers using IQR method from the "age" column and increasing the size of the data (slides 7-8)

4. Scaling the data for building Linear regression models using Standard scaler.

# Q & A

4) What's the purpose of increasing the size of data?

Answer: Having less amount of data makes the model to learn completely on it (in addition to learning the underlying patterns, it also learns the noise in that data), leading to overfit. Means, model may not make predictions accurately on the unseen data. So, to avoid model overfit, one of the methods is to add more data points so that the model we are going to train, can learn the underlying patterns in the data efficiently and can make accurate predictions on the unseen datapoints.

5) Why feature scaling is not necessary for the tree-based models ?

Answer: The tree-based models are not sensitive to the scale of the features. If we consider a decision tree algorithm, it splits a node based on a single feature and this is not influenced by the other features, i.e., there won't be any effect of the remaining features if a split is performed based on one single feature.

6) How was prediction done?

Answer: Based on the performances of each model, I chose XGBoost regressor model. I considered only the top 6 features as per the feature importance by the model and rebuilt it.(slide 12). So, when a user inputs the age of the concrete specimen, quantities of cement, water, fly ash, superplasticizer and the blast furnace slag, the model takes these values as an input and makes a prediction of the compressive strength of the given concrete specimen.



# Q & A

7) What are the different stages of deployment?

Answer: First, deployed the model locally using Flask (a micro web framework) which works as a backend application. The frontend application is a web page designed using HTML5 with CSS styling. So, when a user enters the data and hit "predict" button, model in the backend flask application makes prediction and it will be displayed in the frontend application which user can take a note. Then, deployed this application on web using Heroku and Gunicorn (a python web server gateway interface HTTP server).

8) How are logs managed?

Answer: The entire project is divided into two stages - the development stage and the deployment stage. The logs recorded during the development stage are stored in the "development\_logs.log" file while the logs recorded during the local deployment are stored in the "deployment\_logs.log" file.



Thank you

---