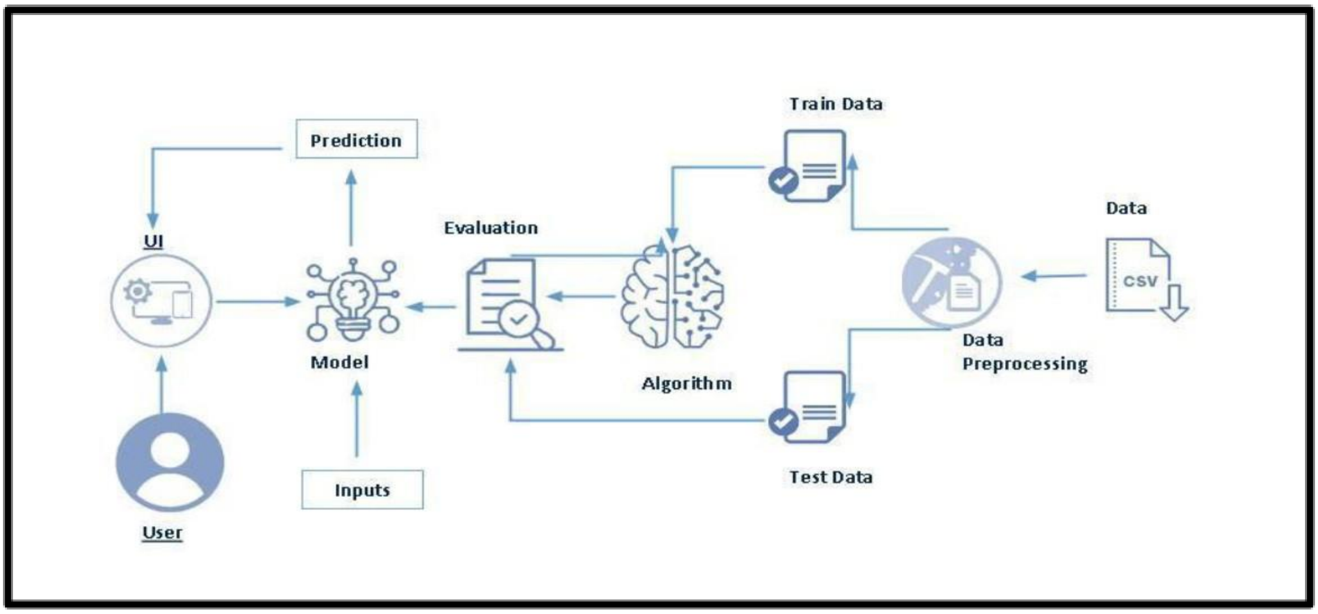
# Doctors Annual Salary Prediction

The goal of this project to predict doctors' Annual salaries using machine learning is to develop a model that can accurately estimate a doctor's salary based on various factors such as their education level, specialty, years of experience, location, and other relevant variables.

This Machine Learning project is to provide useful insights and guidance to doctors and healthcare organizations regarding their compensation structures. By leveraging machine learning techniques, the model can identify the most significant factors that impact doctors' salaries and provide a more accurate salary prediction, which can help doctors negotiate better salaries and help organizations make informed decisions about compensation

## Technical Architecture:



## Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once the model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing
  - Testing model with multiple evaluation metrics
- Model Deployment
  - Save the best model
  - Integrate with Web Framework

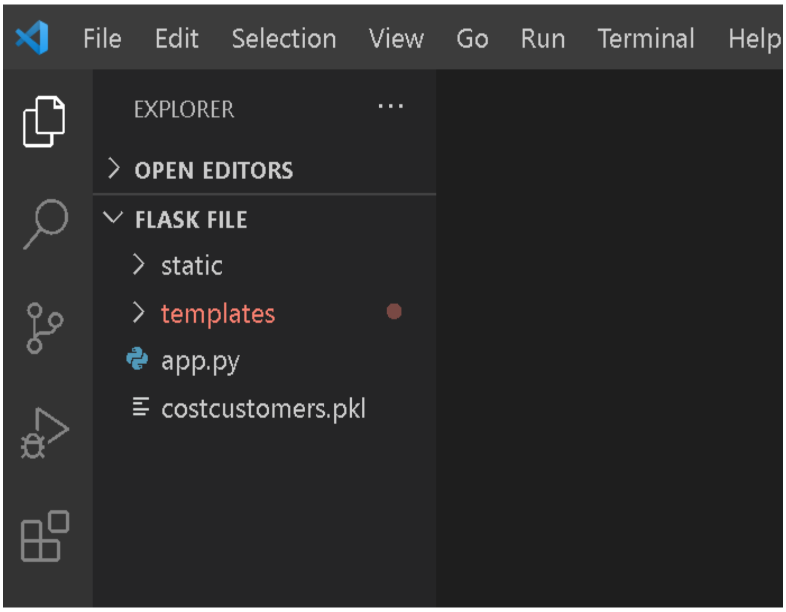
## Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- ML Concepts
    - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
    - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
    - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification- algorithm
    - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
    - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
    - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
    - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
    - NLP:-https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_python.htm
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

## Project Structure:

Create the Project folder which contains files as shown below




- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- startups.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

# Milestone 1: Define Problem / Problem Understanding

## Activity 1: Specify the business problem

Refer Project Description

## Activity 2: Business requirements

To develop a successful cost prediction model for customer acquisition, it's essential to understand the business requirements. Here are some of the key business requirements that need to be considered:

- Accuracy: The model should be able to accurately predict the cost of acquiring customers. This is crucial for businesses to make informed decisions and optimize their marketing campaigns.
- Scalability: The model should be scalable and able to handle large amounts of data. As the business grows, the amount of data will increase, and the model should be able to handle it.
- Real-time predictions: The model should be able to provide real-time predictions so that businesses can make decisions quickly.

## Activity 3: Literature Survey

- A literature survey is an essential step in any machine learning project, as it helps to identify the existing research, techniques, and tools used in the field. Here are some key research papers and articles related to cost prediction for customer acquisition using machine learning: .
- "Using machine learning to predict doctors income: A case study in India" : This study developed a machine learning model to predict physician income based on factors such as age, gender, education, and practice characteristics. The study found that the model had an accuracy rate of 72%, which

## Activity 4: Social or Business Impact.

- Social impact:

  From a social perspective, accurate salary predictions can help ensure that physicians are fairly compensated for their work. By identifying the factors that influence salaries, machine learning models can help address potential biases and disparities in compensation, particularly with regard to demographic factors such as gender and ethnicity

- Business impact:

  From a business perspective, salary predictions can help healthcare organizations better manage their resources and allocate compensation in a more strategic manner. By identifying the factors that influence physician salaries, organizations can develop more effective compensation structures that attract and retain top talent while managing costs

## Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

### Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

**Link:- https://www.kaggle.com/datasets/pythonafroz/doctors**

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

**Note:** There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

### Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
Click here to ask Blackbox to help you code faster
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(random_state=42)
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
✓ 5.8s
```

### Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
df = pd.read_csv("Doctors job dataset.csv")
df.head()
```

## Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

### Activity 2.1: Handling missing values

- Let’s know the info and describe of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used

```
Click here to ask Blackbox to help you code faster
df.info()
df.describe()
✓ 0.1s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 399 entries, 0 to 398
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Work Exp           399 non-null    float64
 1   Annual Salary      399 non-null    float64
 2   Current Location   399 non-null    int32
 3   Designation        399 non-null    int32
 4   U.G. Course        399 non-null    int32
 5   P. G. Course       399 non-null    int32
 6   Age/Date of Birth  399 non-null    float64
dtypes: float64(3), int32(4)
memory usage: 15.7 KB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
Click here to ask Blackbox to help you code faster
df.isnull().sum()
✓ 0.0s

S.No                    0
Resume Title            5
Work Exp                0
Annual Salary           0
Current Location        0
Preferred Location      0
Designation            38
U.G. Course             0
P. G. Course            0
Post P. G. Course     333
Age/Date of Birth      66
Resume ID               0
Last Active Date        0
dtype: int64
```

```
Click here to ask Blackbox to help you code faster
course = 'Post P. G. Course'
df.drop(course, axis=1, inplace=True)
df.tail()
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
df.isnull().mean()*100
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
df['Designation'].mode()
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
df['Designation'].fillna('Consultant Dermatologist', inplace = True
df.isnull().sum()
✓ 0.0s                                                    Python
```

```
Click here to ask Blackbox to help you code faster
df['Resume Title'].fillna('Dermatologist', inplace = True)
df.isnull().sum()
✓ 0.0s                                                    Python
```

```
Click here to ask Blackbox to help you code faster
#filling numerical values with median
df = df.fillna(value = df['Age/Date of Birth'].median())
df.isnull().sum()
✓ 0.0s

S.No                  0
Resume Title          0
Work Exp              0
Annual Salary         0
Current Location      0
Preferred Location    0
Designation           0
U.G. Course           0
P. G. Course          0
Age/Date of Birth     0
Resume ID             0
Last Active Date      0
dtype: int64
```

### Activity 2.2: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using Label encoding with the help of list comprehension.

- In our project, categorical features are in many columns Label encoding is done

```
Click here to ask Blackbox to help you code faster
df['Current Location'] = le.fit_transform(df['Current Location'])
df['Designation'] = le.fit_transform(df['Designation'])
df['U.G. Course'] = le.fit_transform(df['U.G. Course'])
df['P. G. Course'] = le.fit_transform(df['P. G. Course'])
df.tail()
```

0.0s Python

## Milestone 3: Exploratory Data Analysis

### Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

### Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

### Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature.

.

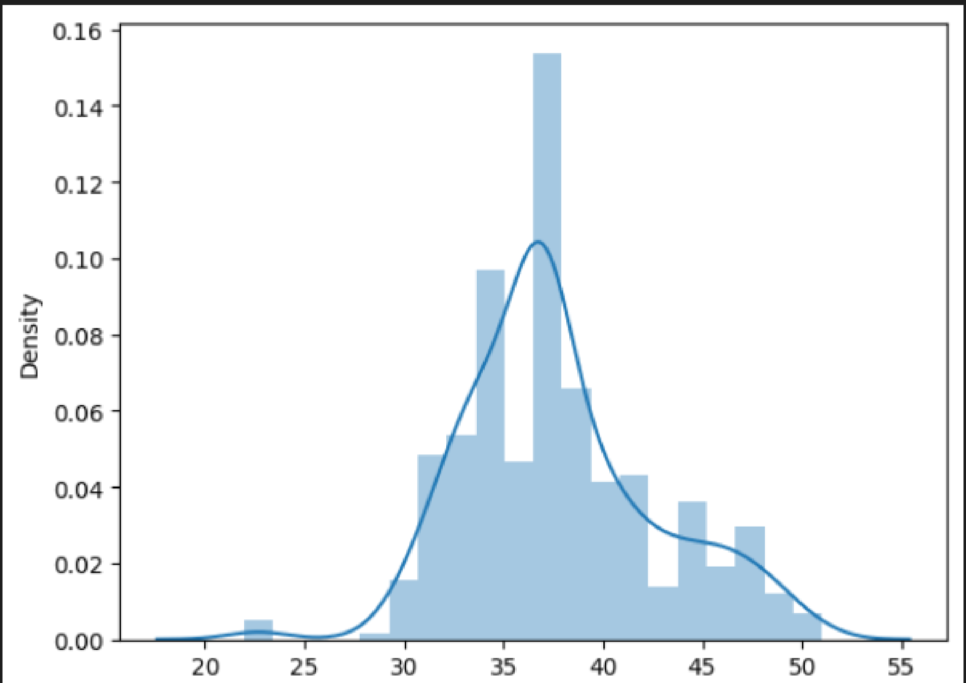
UNIVARIATE ANALYSIS

💡 Click here to ask Blackbox to help you code faster

```
import seaborn as sns
sns.distplot(df['Age/Date of Birth'])
```

✓ 1.5s

```
C:\Users\AMIT KUMAR BHADRA\AppData\Local\Temp\ipykernel_3648\3867061911.py:2: UserWarning

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Age/Date of Birth'])

<Axes: xlabel='Age/Date of Birth', ylabel='Density'>
```

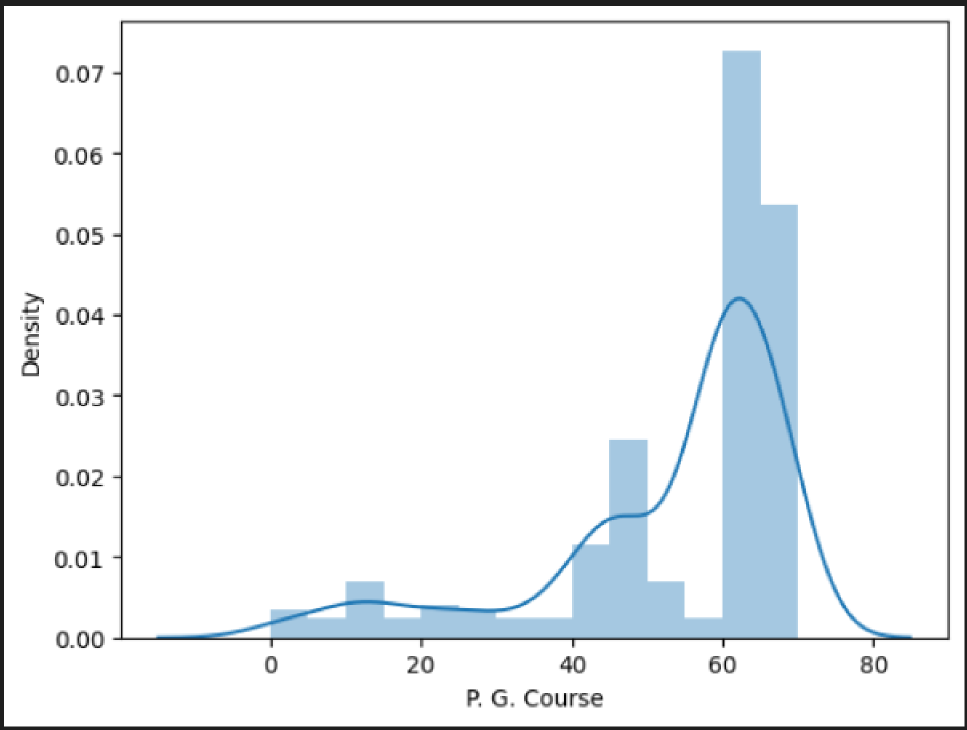


Click here to ask Blackbox to help you code faster

```
sns.distplot(df['P. G. Course'])
```

0.5s

```
C:\Users\AMIT KUMAR BHADRA\AppData\Local\Temp\ipykernel_3648\2409161705.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['P. G. Course'])

<Axes: xlabel='P. G. Course', ylabel='Density'>
```



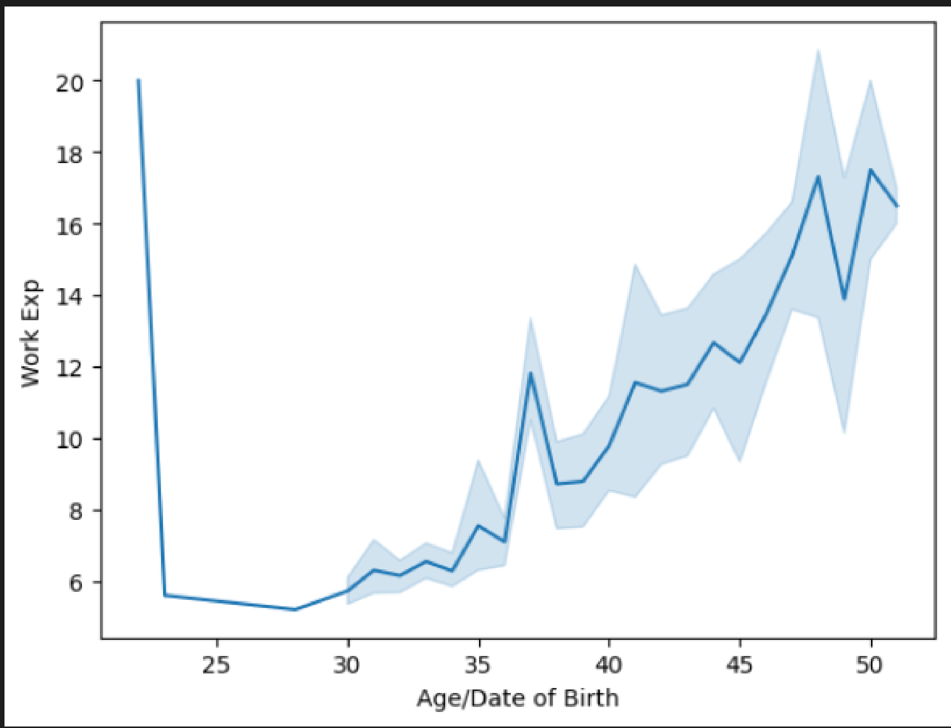
## Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis.
Here we are visualizing.

BIVARIATE ANALYSIS

```
Click here to ask Blackbox to help you code faster
sns.lineplot(x = df['Age/Date of Birth'], y =df['Work Exp'])
✓ 1.1s
```

```
<Axes: xlabel='Age/Date of Birth', ylabel='Work Exp'>
```




## Splitting data into train and test

Now let’s split the Dataset into train and test sets. First split the dataset into x and y and then split the data set  Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data ware using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

SPLITTING INTO X AND Y

```
Click here to ask Blackbox to help you code faster
X = df.drop('Annual Salary', axis =1)
y = df['Annual Salary']
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 42)
✓ 0.0s
```

## Milestone 4: Model Building

### Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project, we are applying three classification algorithms. The best model is saved based on its performance.

### Activity 1.1: Linear Regression

A function named Linear Regression is created and train and test data are passed as the parameters. Inside the function, Linear Regression algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model with R2_score.

LINEAR REGRESSION

```
Click here to ask Blackbox to help you code faster
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
reg.fit(X_train, y_train)
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
y_train_pred = reg .predict(X_train)
y_test_pred = reg .predict(X_test)
y_train_pred[:5], y_test_pred[:5]
✓ 0.0s
```

Checking the accuracy for linear reg

```
Click here to ask Blackbox to help you code faster
r2_score(y_train, y_train_pred)*100 , r2_score(y_test, y_test_pred)*100
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
mean_squared_error(y_train,y_train_pred), mean_squared_error(y_test,y_test_pred)
✓ 0.0s
```

### Activity 1.2: Random Forest Regressor

A function named random forest regressor is created and train and test data are passed as the parameters. Inside the function, random forest regressor algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model with R2_score.

```
Click here to ask Blackbox to help you code faster
rf.fit(X_train,y_train)
0.3s
```

```
Click here to ask Blackbox to help you code faster
y_test_pred = rf.predict(X_test)
y_train_pred = rf.predict(X_train)
y_train_pred, y_train_pred
0.0s
```

```
Click here to ask Blackbox to help you code faster
acc = r2_score(y_test,y_test_pred)
acc
0.0s
```

```
Click here to ask Blackbox to help you code faster

mae = mean_absolute_error(y_test, y_test_pred)
mae
0.0s
```

```
Click here to ask Blackbox to help you code faster
rf.predict([[12.0,26,159,0,60,37.0]])
0.0s
```

### Activity 1.3: Decision Tree Regressor

A function named decision Tree regressor is created and train and test data are passed as the parameters. Inside the function, decision Tree regressor algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, For evaluating the model with R2_score

# DECISION TREE REGRESSOR

```
Click here to ask Blackbox to help you code faster
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(random_state =42)
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
dtr.fit(X_train, y_train)
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
y_train_pred = dtr.predict(X_train)
y_test_pred = dtr.predict(X_test)
y_train_pred[:5], y_test_pred[:5]
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
r2_score(y_train, y_train_pred)*100, r2_score(y_test, y_test_pred)*100
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
mean_squared_error(y_train, y_train_pred), mean_squared_error(y_test, y_test_pred)
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
dtr.predict([[7.0,34,44,0,60,37.0]]) # decision tree
✓ 0.0s

c:\Users\AMIT KUMAR BHADRA\AppData\Local\Programs\Python
  warnings.warn(

array([18.5])
```

### Activity 1.4: Gradient Boosting Regressor

A function named Gradient boosting regressor is created and train and test data are passed as the parameters. Inside the function, Gradient boosting regressor algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, For evaluating the model with R2_score

# XGBOOST

```
Click here to ask Blackbox to help you code faster
import xgboost as xgb
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
xg = xgb.XGBRegressor()
xg.fit(X_train,y_train)
✓ 0.1s
```

```
Click here to ask Blackbox to help you code faster
y_train_pred = xg.predict(X_train)
y_test_pred = xg.predict(X_test)
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
r2_score(y_train,y_train_pred)*100 , r2_score(y_test,y_test_pred)*100
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
mean_squared_error(y_train,y_train_pred), mean_squared_error(y_test,y_test_pred)
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
xg.predict([[7.0,34,44,0,60,37.0]]) # xg boost
✓ 0.0s

array([17.6803], dtype=float32)
```

### Activity 2: Testing the model

Here we have tested with Logistic regression and Svm algorithms. With the help of predict() function.

```
Click here to ask Blackbox to help you code faster
rf.predict([[7.0,34,44,0,60,37.0]]) # random forest regressor
✓ 0.0s

c:\Users\AMIT KUMAR BHADRA\AppData\Local\Programs\Python\Python39\l
  warnings.warn(

array([15.309])
```

```
Click here to ask Blackbox to help you code faster
reg.predict([[7.0,34,44,0,60,37.0]]) # linear regression
✓ 0.0s

c:\Users\AMIT KUMAR BHADRA\AppData\Local\Programs\Python\Python39\l
  warnings.warn(

array([11.51278812])
```

```
Click here to ask Blackbox to help you code faster
dtr.predict([[7.0,34,44,0,60,37.0]]) # decision tree
✓ 0.0s

c:\Users\AMIT KUMAR BHADRA\AppData\Local\Programs\Python\Python39\l
  warnings.warn(

array([18.5])
```

```
Click here to ask Blackbox to help you code faster
xg.predict([[7.0,34,44,0,60,37.0]]) # xg boost
✓ 0.0s

array([17.6803], dtype=float32)
```

# Milestone 5: Model Deployment

### Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
Click here to ask Blackbox to help you code faster
import pickle
✓ 0.0s
```

```
Click here to ask Blackbox to help you code faster
pickle.dump(dtr,open('startups.pkl','wb'))
✓ 0.0s
```

### Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

### Activity 2.1: Building Html Pages:

### Activity 2.2: Build Python code:

Import the libraries in python file

```
from flask import Flask,render_template,request
import pickle
import numpy as np
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (_name__) as argument.

```
app = Flask(__name__)
model = pickle.load(open("startups.pkl","rb"))
```

Render HTML page:

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

```
@app.route("/")
def start():
    return render_template('index.html')
```

In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.
Retrieves the value from UI:

```
@app.route('/login',methods=['POST'])

def login():
    p = request.form['we']
    q = request.form['cl']
    r = request.form['deg']
    s = request.form['ug']
    t = request.form['pg']
    u = request.form['age']



    t= [[float(p),float(q),float(r),float(s),float(t),float(u)]]
    output = model.predict(t)
    print(output)

    return render_template("index.html",y="PREDICTED ANNUAL SALARY: "+str((output[0]))+" LPA")

if __name__ == '__main__' :
    app.run(port = 5000,debug=True)
```

Here we are routing our app to prediction() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```
if __name__=='__main__':
    app.run(debug=True)
```

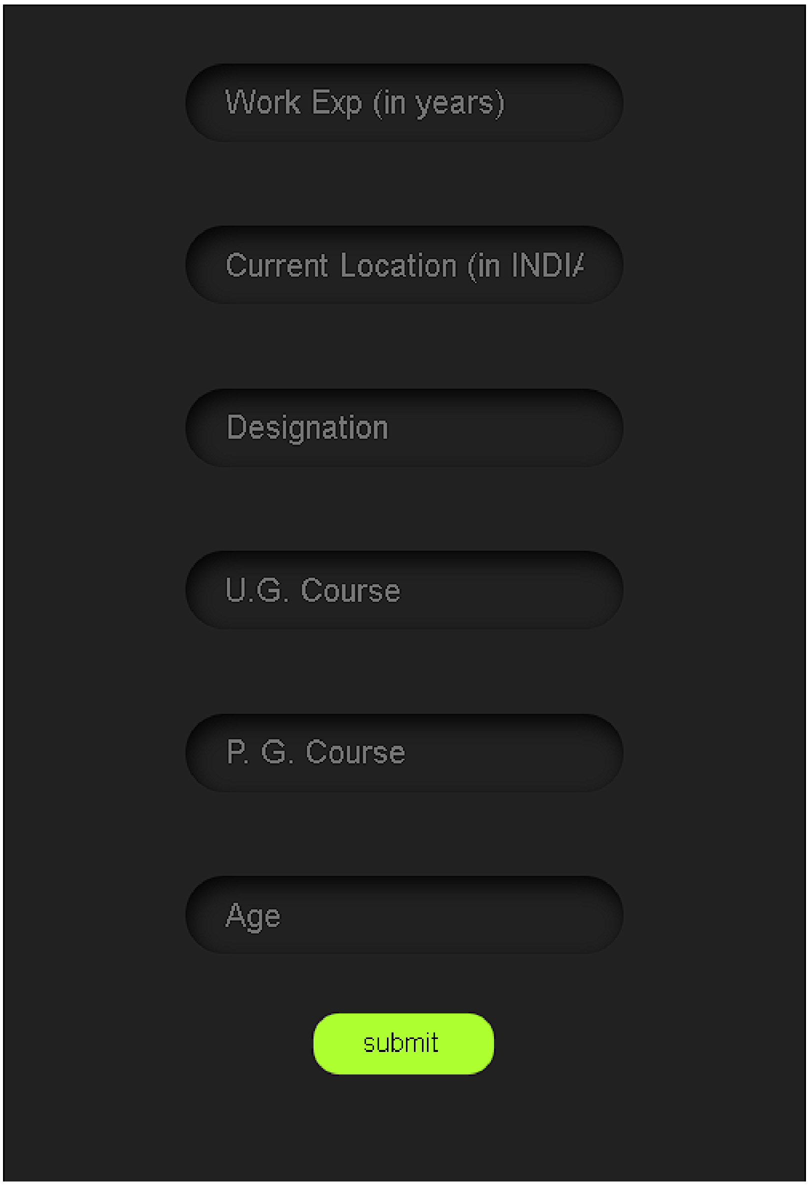
## Activity 2.3: Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
[Running] python -u "c:\Users\AMIT KUMAR BHADRA\Documents\Machine Learning\Project\Project Development Phase\apps.py"
 * Serving Flask app 'apps'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 760-498-314
```

Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result.  Now Model Predicting Annual Salary Of The Doctors

# DOCTOR'S ANNUAL SALARY PREDICTOR



Now Showing Result