



# Insurance Premium Prediction

---

Vaibhav Joshi



## **Objective:**

Development of a predictive Regressor model to find premium of insurance one has to pay depending on conditions mentioned. This model requires age, No of children, smoker or no, bmi, region and sex of individual respectively.

## **Benefits:**

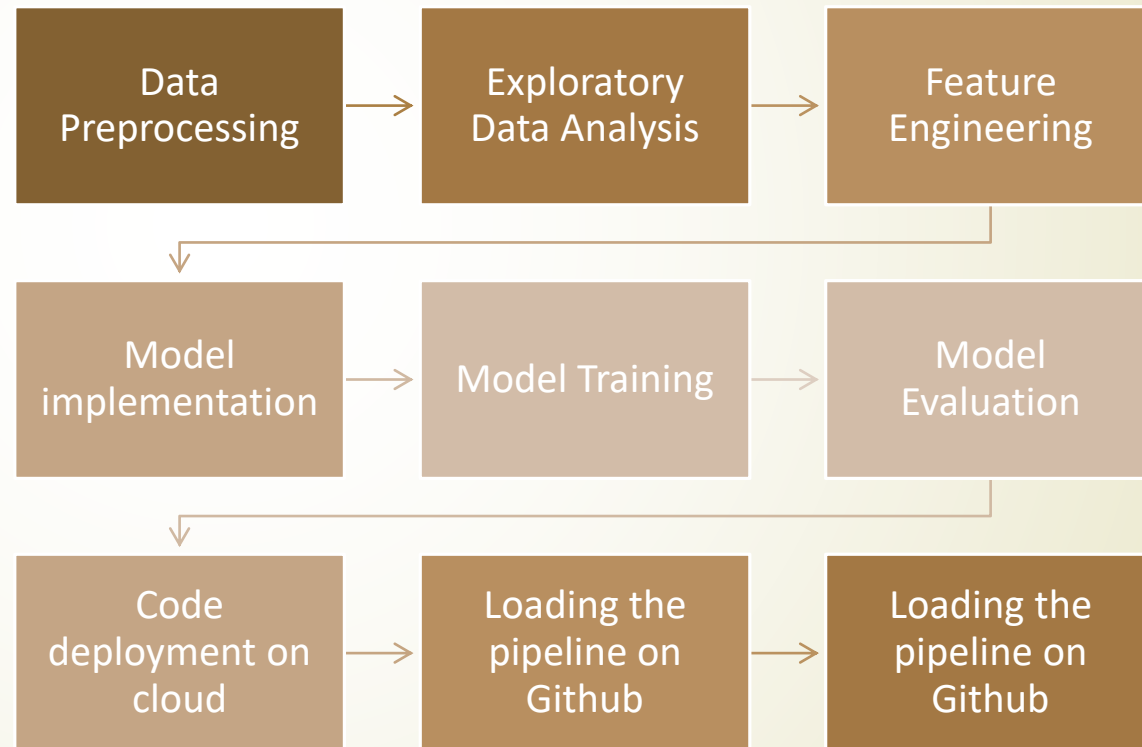
This model can be used to determine the expenses on premium one have to do and can be used to help individual to plan according to that.

# Architecture

## Data Preparation

## Model development

## Deployment

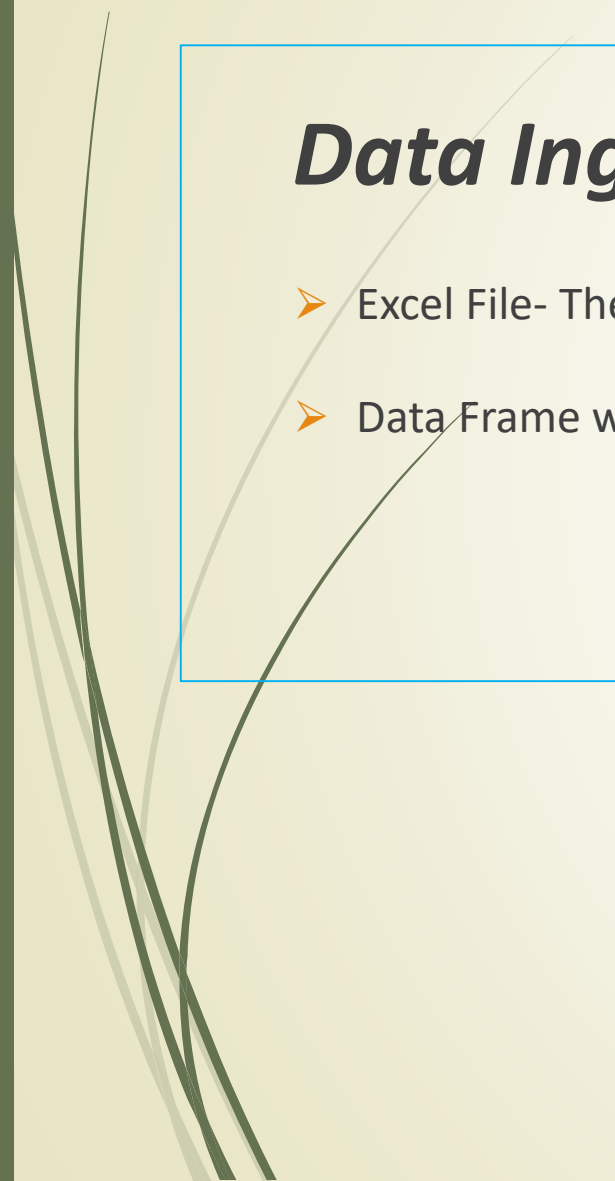


# Data validation and transformation

DATA TYPE	NULL VALUES	NUMERICAL COLUMNS	CATEGORICAL COLUMNS
Data type of columns is given in the schema file. It is validated when we insert the files into Database.	If any of the columns in a file have all the values as NULL or missing, we can fill it by some methods.	All the numerical features were standardized using Standard Scaler, preventing any data leakage.	Ordinal Encoding was used to treat categorical columns for the model in understandable way.
If data type is wrong, we can convert it using pandas library.	We can fill them by using mode of categorical columns or mean of numerical columns.	This process is done in pipeline for numerical features for the convenience of deployment.	This process is done in pipeline for categorical features for the convenience of deployment.



## ***Data Ingestion***

- Excel File- The dataset was imported from Excel File into python.
  - Data Frame was created using pandas.
- 



# ***Model Training***

1. The data in database is imported to Jupyter notebook by using pandas.
2. In data preprocessing step, data is checked if there missing data, duplicate values, and datatypes of each feature. In our dataset, there was not any null and duplicate values.
3. After train and test splitting, pipeline containing Standard Scaler and Ordinal Encoder was fitted to several models.





## ***Model Selection:***

Having trained several models and obtained R2 scores, it was determined that Gradient Boosting performs better than other models.

## ***Prediction***

The model is made in such a way to maximize the accuracy and also other performance metrics so that the predictions are as accurate as possible.

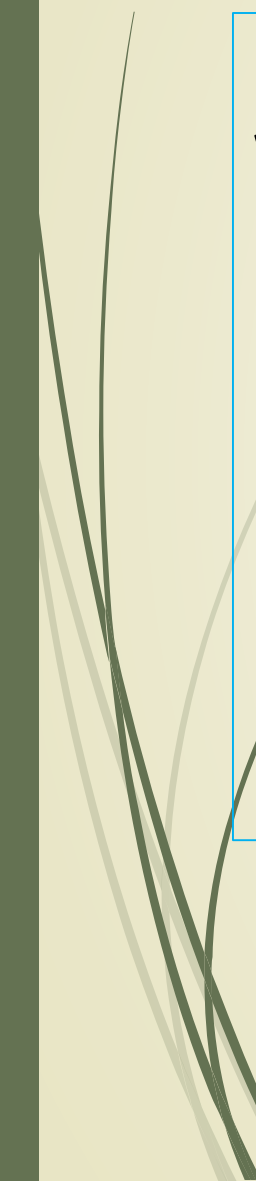
# Q&A

What is the source of data?	What was the type of data?	What is the complete flow you followed in this project?	How are logs managed?
The data for training is provided by the client in the form answers to certain questions asked . User has to enter answers for those questions.	The data is the combination of both numerical and categorical values.	Refer to 3 <sup>rd</sup> slide for the process flow.	Following s are the logs that we are using :  Data Insertion log, Model Fitting log, prediction log, etc.





## What techniques were you using for data pre-processing?

- Removing unwanted attributes
  - Visualizing relation of independent variables with each other and output variables
  - Checking and changing Distribution of continuous values
  - Cleaning data and imputing if null values are present.
  - Converting categorical data into numeric values.
  - Scaling the data
- 



## How training was done or what models were used?

- First, we started with data cleaning, EDA and feature engineering. Data type of columns were corrected by using pandas attributes.
- Ordinal encoding, and numerical columns was scaled using Standard Scaler.
- Data pipeline was created to implement data scaling, Ordinal encoding and an estimator to prevent any data leakage.
- Gradient Boosting model was used as the best estimator.



## How was prediction done?

Some questions were asked to the user like age, Sex, BMI, Region, No of children and smoker or no and these responses

are taken as inputs which are then feed to the model as a single test case and the premium prediction are then returned

on the screen after a few seconds in which the data pipeline processes the input data to get the output.



## What are the different stages of deployment?

- A server is created in Flask which just displays a single web page and runs the uplink code via threading thereby creating a server.
- The code was pushed to Github.
- The pipeline was created on AWS cloning with the Github.
- Then the project was deployed on the AWS.