

ARCHITECHURE DESIGN

# Insurance Premium Prediction

NIKITA MAKARAND PANDE

# Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Date Issued** | **Version** | **Description** | **Author** |
| **01.01.2023** | V1.0 | Initial LLD- V1.0 | NIKITA |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **Content** | **Page No** |
| Abstract | 4 |
| 1. Introduction | 4 |
| 1.1 What is Architecture Design? | 4 |
| 1.2 Scope | 4 |
| 1.3 Constraints | 4 |
| 2. Technical Specification | 5 |
| 2.1 Dataset | 6 |
| 2.2 Logging | 6 |
| 2.3 Deployment | 6 |
| 3. Technology Stack | 7 |
| 4. Proposed Solution | 7 |
| 5. Architecture | 7 |
| 5.1 Architecture Description | 8 |
| 6. User Input/Output Workflow | 10 |

**INDEX**

## ABSTRACT

Machine Learning is a category of algorithms that allows software applications to become more accurate in predicting outcomes without being explicitly program. The basic premise of machine learning is to build models and employ algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. These models can be apply in different areas and trained to match the expectations of management so that accurate steps can be taken to achieve the organization’s target. In this project, we will estimate the amount of insurance premium based on personal health information. Taking various aspects of a dataset collected from people, and the methodology followed for building a predictive model.

**1. Introduction**

**1.1 What is Architecture Design?**

The primary goal of Architecture Design (AD) is to provide the internal design for the `Insurance Premium Prediction` program. AD documents class diagrams and specifies how classes relate to one another, as well as specifying which modules should be write by the programmer in order to create this program.

**1.2 Scope**

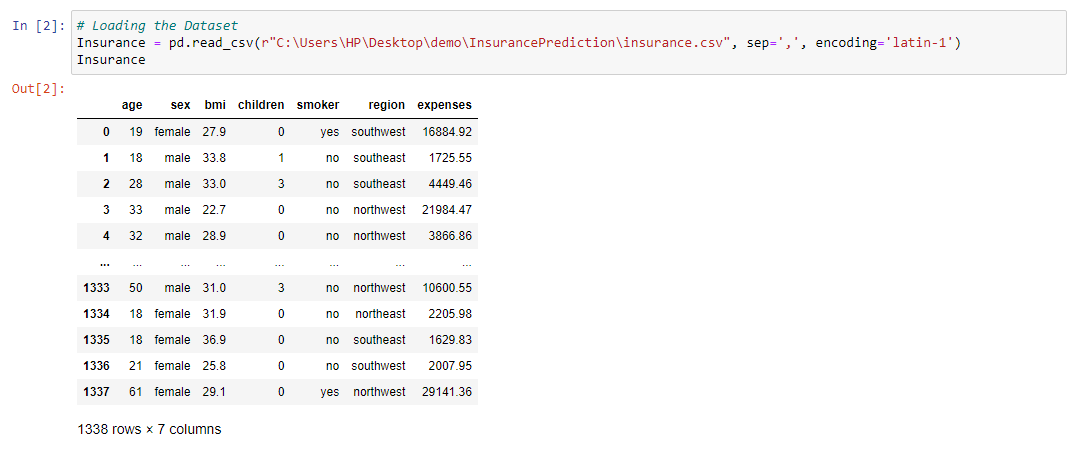
Architecture design is a step-by-step process that can be used to create data structures, required software, architecture, source code, and ultimately performance algorithms. The refinement of the overall data organization can occur during requirement analysis; once this has been done, design work may commence. Overall flow: Requirement Analysis -> Data Design -> Implementation

**2. Technical Specification**

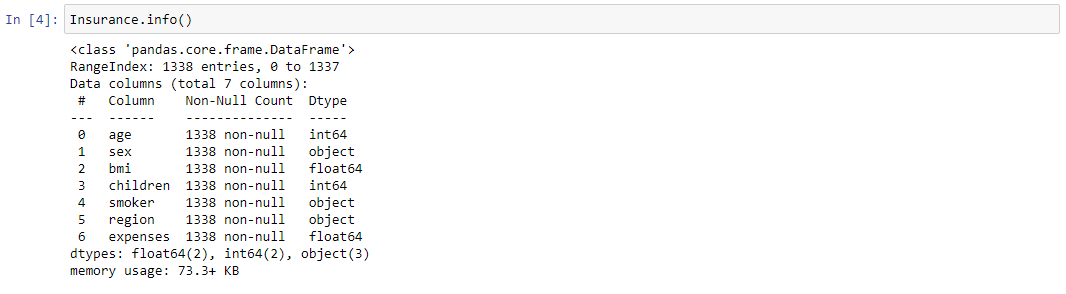
**2.1 Dataset**

The dataset containing verified historical data, consisting of the aforementioned information and the actual medical expenses incurred by over 1300 customers. The objective is to find a way to estimate the value in the "expenses" column using the values in the other columns like their age, sex, BMI, no. of children, smoking habits and region. Using all the observations it is inferred what role certain properties of user. how they affect their expenses.

The dataset looks like as follow:

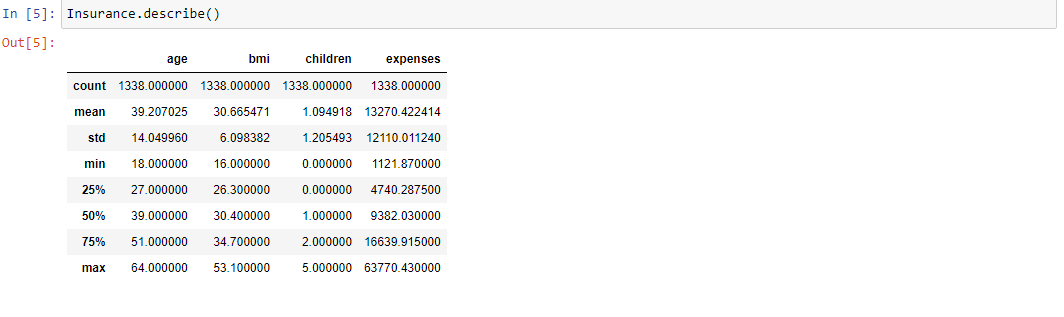


The data set consists of various data types from integer to floating to object as shown in Fig.

****

In the dataset, there can be various types of underlying patterns which also gives an in-depth knowledge about the subject of interest and provides insights into the problem. Looks like ‘age’, ‘children’, ‘bmi’ (body mass index) and ‘expenses’ are numbers, whereas ‘sex’, ‘smoker’, and ‘region’ are strings (possibly categories).

Various factors important by statistical means like mean, standard deviation, median, count of values and maximum value, etc. are shown below for numerical attributes

****

**2.2 Logging**

We should be able to log every activity done by the user

• The system identifies at which step logging require.

• The system should be able to log each and every system flow.

• The system should be not be hung even after using so much logging. Logging just because we can easily debug issuing so logging is mandatory to do.

**2.3 Deployment**

For the hosting of the project, we will use Heroku and GitHub platform.





**3. Technology Stack**

|  |  |
| --- | --- |
| **Front End** | HTML/CSS |
| **Backend** | Python/ Flask |
| **Deployment** | Heroku /GitHub |

**4. Proposed Solution**

We will use performed EDA to find the important relation between different attributes and will use a machine-learning algorithm to estimate the cost of expenses. The client will be fill the required feature as input and will get results through the web application. The system will get features and it will be passed into the backend where the features will be validated and preprocessed and then it will be passed to a hyper parameter tuned machine-learning model to predict the final outcome.

**5. Architecture**

End

Model Building

Visualization

Start

Deployment

Model Testing

Data Cleaning

Export data from csv

Flask Setup

Prediction

Data Transformation

Data Preprocessing

**5.1 Data Gathering**

Data source: https://www.kaggle.com/noordeen/insurance-premium-prediction Dataset is stored in .csv format.

**5.2 Raw Data Validation**

After data is loaded, various types of validation are required before we proceed further with any operation. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because the attributes, which contain these, are of no use. It will not play role in contributing to the estimating cost of the premium.

**5.3 Exploratory Data Analysis**

Visualized the relationship between the dependent and independent features. Also checked relationship between independent features to get more insights about the data.

**5.4 Feature Engineering**

After pre-processing standard scalar is performed to scale down all the numeric features. Even one hot encoding is also performed to convert the categorical features into numerical features. For this process, pipeline is created to scale numerical features and encoding the categorical features.

**5.5 Model Building**

After doing all kinds of pre-processing operations mention above and performing scaling and encoding, the data set is passed through a pipeline to all the models, Linear Regression, Decision tree, Random Forest and Gradient booster. It was found that Gradient boosting performs best with the on test data after that we perform Gradient SearchCV on gradient booster to find out best parameters, So after ‘Gradient SearchCV’ on gradient booster our model performance increases.

**5.6 Model Saving**

Model is save using dill library in .pkl format.

**5.7 Flask Setup for Web Application**

After saving the model, the API building process started using Flask. Web application creation was create in Flask for testing purpose. Whatever user will enter the model to estimate the premium of insurance will extract the data and then that data; this is perform in this stage.

**5.8 GitHub**

The whole project directory will be push into the GitHub repository.

**5.9 Deployment**

The project was deploy from GitHub into the Heroku platform.

**6. User Input / Output Workflow.**

**Start Python File**

**Submit Details**

**Insert User Input**

**Start Python File**

**Preprocess and clean the data**

**Predict The Result**