Insurance Premium Prediction

Low Level Design

Documentation (LLD).

Internship Project For- iNeuron (Insurance Premium Prediction)

Done By – Md Rian Ur Rahaman

Contents

1. Abstract

This data science project focuses on developing a comprehensive solution to predict and mitigate customer churn in a subscription-based business model. The project follows a systematic end-to-end approach, encompassing data collection, preprocessing, exploratory data analysis (EDA), feature engineering, model selection, training, and deployment. The primary objective is to empower businesses with actionable insights to retain valuable customers and optimize revenue streams.

The project begins with data collection from diverse sources, including customer interactions, transaction history, and demographic information. Data preprocessing involves cleaning, handling missing values, and transforming raw data into a format suitable for analysis. The EDA phase employs statistical and visual techniques to uncover patterns, correlations, and potential factors influencing customer churn.

The project emphasizes the importance of interpretability, providing stakeholders with clear insights into the factors contributing to customer churn. Model explanations and visualizations aid in understanding the decision-making process and guide strategic decision-making.

The selected model is then deployed into a production environment, allowing real-time predictions on new data. Integration with existing systems and continuous monitoring ensure the model's effectiveness over time.

The project concludes with a comprehensive evaluation of the entire process, including model performance metrics, business impact, and insights gained.

By implementing this end-to-end data science project, businesses can proactively address customer churn, reduce revenue loss, and enhance overall customer satisfaction.

2. Introduction.

2.1. What is a Low-Level design document?

The goal of LLD or a low-level design document (LLD) is to give the internallogical design of the actual program code for Rental Bikeshare Demand Prediction.

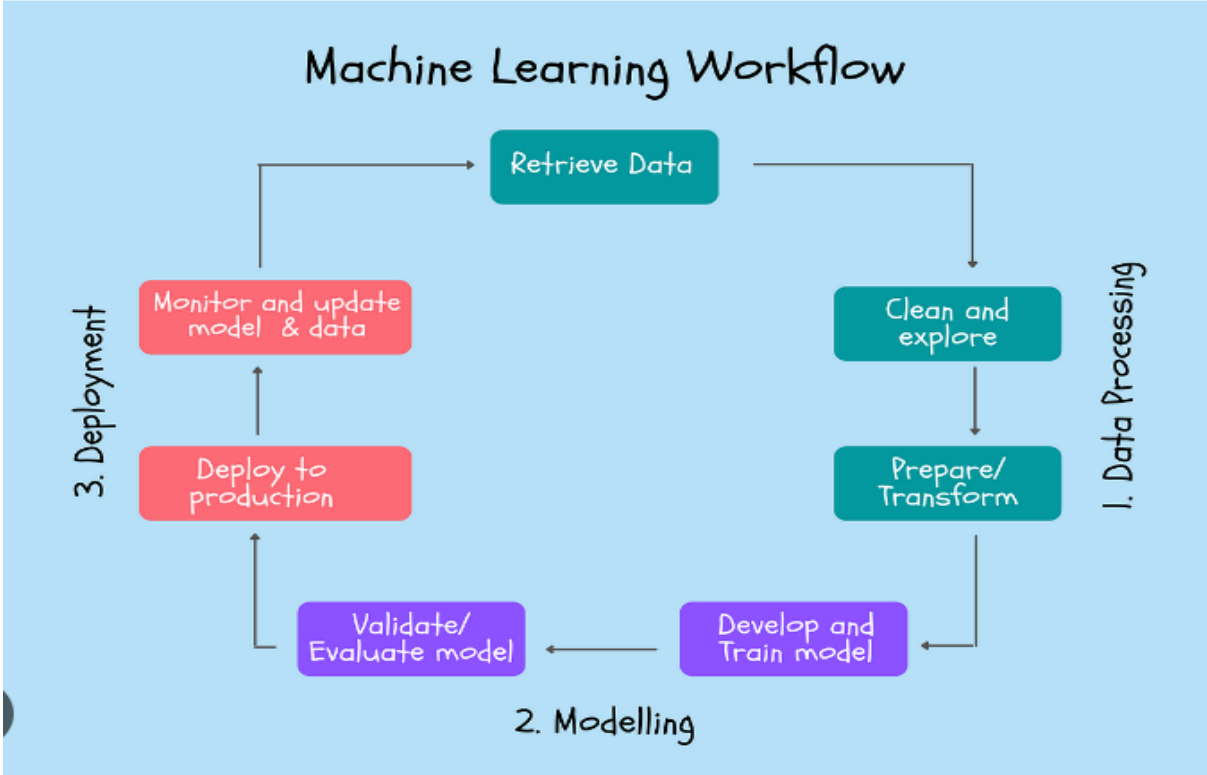
LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

2.2. Scope of (LLD)

Low-level design (LLD) is a component-level design process that follows a

step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

3. Architecture (Machine learning project workflow).



3.1 ARCHITECTURE DESIGN

This project is completely based on the life cycle of machine learning, where we will be predicting the count of rental bikes that would be needed based on certain conditions. The tools used in this project are Python, Pandas, NumPy, Matplotlib,Seaborn, Scikit learn, Flask for Backend, HTML, CSS & Java script was used for the Frontend part of the web application.

3.2 Data Requirement. Whenever we are working on any project the data is completely dependent on the requirement of the problem statement. For this project the problem statement was to create a Hyper tuned Regression machine learning model which can predict the total count of bikes rented at a particular day based on various parameters.

3.3 Data Collection.

The data (hour.csv file) which is used in this project was taken from UCI

Machine Learning Repository.

DataSet Link : https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction

3.4 About The Dataset :

Context

The insurance.csv dataset contains 1338 observations (rows) and 7 features (columns). The dataset contains 4 numerical features (age, bmi, children and expenses) and 3 nominal features (sex, smoker and region) that were converted into factors with numerical value designated for each level.

Inspiration

The purposes of this exercise to look into different features to observe their relationship, and plot a multiple linear regression based on several features of individual such as age, physical/family condition and location against their existing medical expense to be used for predicting future medical expenses of individuals that help medical insurance to make decision on charging the premium.

The insurance.csv dataset contains 1338 observations (rows) and 7 features (columns). The dataset contains 4 numerical features (age, bmi, children and expenses) and 3 nominal features (sex, smoker and region) that were converted into factors with numerical value desginated for each level.

The purposes of this exercise to look into different features to observe their relationship, and plot a multiple linear regression based on several features of individual such as age, physical/family condition and location against their existing medical expense to be used for predicting future medical expenses of individuals that help medical insurance to make decision on charging the premium.

3.6 Tools / Software Used:-

● Python version used for this project 3.9.0 (This may get updated and some features might not be available in new version.)

● Python libraries such as NumPy, pandas, matplotlib, seaborn and scikit-learn (Used for implementation of machine learning algorithms)

● Jupyter for Exploratory Data Analysis and testing code, Visual studio code is used as an IDE for writing the code.

● HTML, CSS & Java Scripts are used for developing the front end of our web application.

● Flask is used for backend development.

● Github is used as the version control system.

Github Link : https://github.com/mdrianurrahaman/Insurence

3.7 Loading data.

Local file system was used for loading the dataset (hour.csv) using read_csv function of Pandas Library

3.8 Data Preprocessing:

To preprocess this dataset, which contains information about individuals' age, sex, BMI, number of children, smoking status, region, and medical expenses, several steps are necessary to prepare it for analysis or modeling. First, we'll handle any missing values to ensure the dataset's completeness, as missing data can impact model performance. Next, we'll convert categorical variables, such as 'sex', 'smoker', and 'region', into numerical formats using techniques like one-hot encoding. This step is crucial for algorithms that require numerical input. For instance, 'sex' will be transformed into binary variables, while 'region' and 'smoker' can be encoded into separate columns.

We will also normalize or standardize the numerical features, such as 'age', 'BMI', and 'expenses', to ensure they are on a similar scale, which can improve the performance of many machine learning algorithms. Additionally, the 'children' column, which contains count data, will be left as is since it is already numerical. After these transformations, we will split the data into training and testing sets to evaluate the model's performance effectively. If necessary, feature engineering techniques may be applied to create new features that could enhance the model's predictive power. Finally, we will ensure that the data is ready for input into various machine learning models, aiming to predict the 'expenses' based on the other attributes. This comprehensive preprocessing will help build a robust model and derive meaningful insights from the data.

3.9 Model Training :

The provided code snippet demonstrates the process of training and selecting the best model for predicting insurance premiums. The code begins by importing essential libraries, including various regression models from sklearn, custom exception handling, logging functionalities, and utility functions for saving objects and evaluating models. The ModelTrainerConfig dataclass is used to specify the file path for saving the trained model. The ModelTrainer class is then defined, initializing the configuration and providing the initiate_model_training method.

In the initiate_model_training method, the training and testing datasets are split into dependent (y) and independent (X) variables. A dictionary of regression models is created, including LinearRegression, DecisionTreeRegressor, RandomForestRegressor, AdaBoostRegressor, SGDRegressor, Lasso, ElasticNet, SVR, and Ridge. The evaluate_model function is then called to evaluate each model's performance on the training and testing data, and the results are logged and printed.

The method identifies the best-performing model based on the highest R2 score from the evaluation report. The name and score of the best model are logged and printed, highlighting the best model found. Finally, the best model is saved to the specified file path using the save_object utility function. Exception handling ensures that any errors during model training are logged, and a custom exception is raised. This structured approach ensures robust model training and selection, facilitating effective prediction of insurance premiums.

3.10 UI Integration

As stated in the problem statement we have to create a user interface page where users

can input their data and our machine learning model will give them the predicted result.

3.11 Data Taken from the user

Data from the user is retrieved using the HTML & CSS web app, and output is calculated using the app.py as our Flask application which is using our machine learning model to give the predicted result.

3.12 Data Validation

The data which is entered by the user is validated by the app.py file which is built using the python flask and then this data is transferred to our model.

3.13 Rendering the result

The result for our model is rendered at the HTML page. As we created an API for our predicted result page.