Low Level Document

ADULT CENSUS INCOME PREDICTION

LLD History :-

Date	Version	Author
22/09/2021	V1.2	Umang Tank

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

Abstract

- Census data had an aim to increase the awareness about how the income factor actually has an impact not only on the personal lives of people, but also an impact on the nation and its betterment. We will today have a look on the data extracted from the 1994 Census bureau database, and try to find insights about how different features have an impact on the income of an individual.

- Though the data is quite old, and the insights drawn cannot be directly used for derivation in the modern world, but it would surely help us to analyze what role different features play in predicting the income of an individual.

1. Introduction

1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Back Order Prediction Model. 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.

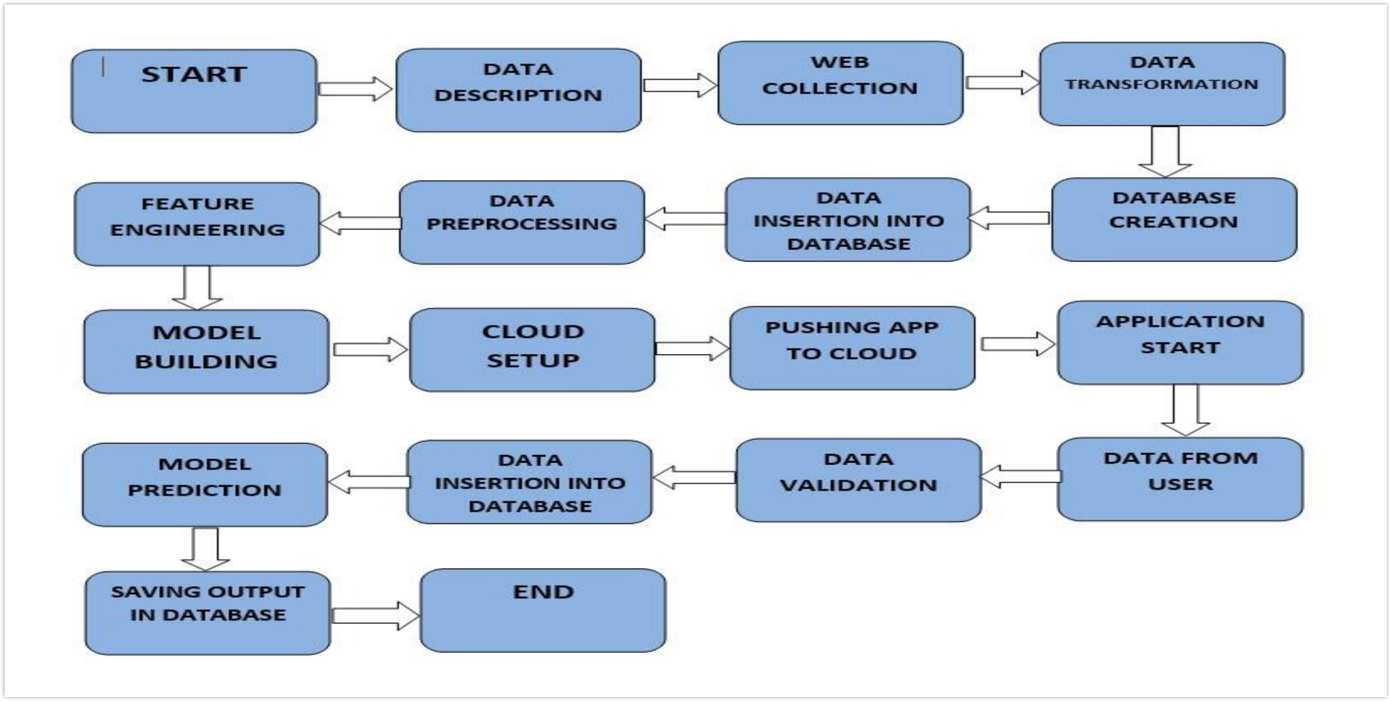
1.2. Scope

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.

1.3. Constraints

Internet connection is a constraint for the application. Since the application fetched the data from the database, it is crucial that there is an Internet connection for the application to function. Since the model can make multiple requests at same time, it may be forced to queue incoming requests and therefore increase the time it takes to provide the response.

2.Architecture



3.Architecture Description

3.1. Data Description

The dataset provided to us contains many rows, and 13 different independent features. We aim to predict if a person earns more than 50k$ per year or not. Since the data predicts 2 values (>50K or <=50K), this clearly is a classification problem, and we will train the classification models to predict the desired outputs.

Age — The age of an individual, this ranges from 17 to 90.

Workclass — The class of work to which an individual belongs.

Education — Highest level of education

Education_num — Number of years for which education was taken

Marital_Status — Represents the category assigned on the basis of marriage status of a person

Occupation — Profession of a person

Relationship — Relation of the person in his family

Race — Origin background of a person

Sex — Gender of a person

Capital_gain — Capital gained by a person

Capital_loss — Loss of capital for a person

Hours_per_week — Number of hours for which an individual works per week

Native_Country — Country to which a person belongs

Income — The target variable, which predicts if the income is higher or lower than 50K$.

3.2. Data Cleaning / Data Transformation

In the Cleaning process, we have cleaned up all the data because data is present in very bad format which was can not recognized by machine (Categorical data). So data Cleaning is done very first by data validation methods. Get_dummies method use for Convert Categorical data in numerical format.

3.3. Exploratory Data Analysis

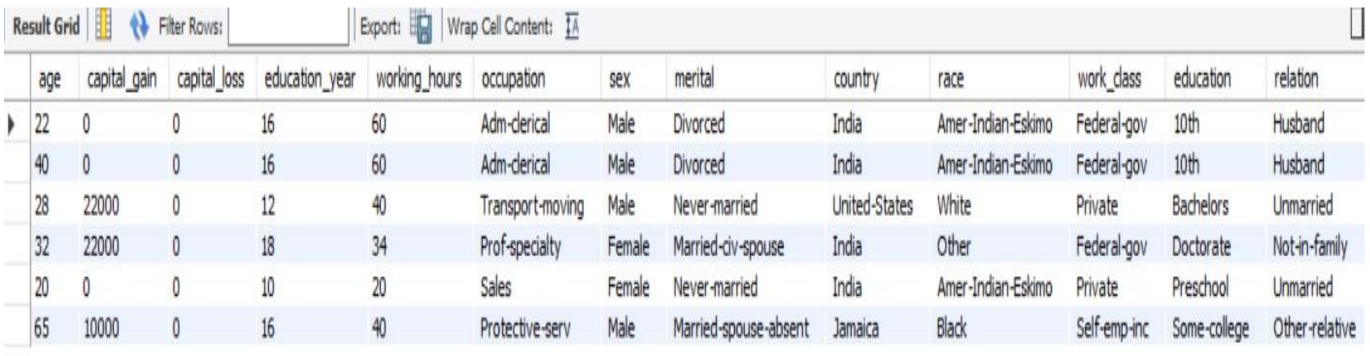
In EDA we have seen various insights from the data so we have selected which column is most important and dropped some of the columns by observing them plotting their heatmap from seaborn library also we done null value managed in an efficient manner and also implemented categorical to numerical transfer of column method here.

3.4. Data Insertion into Database

Database Creation and connection - Create a database with name passed. If the database is already created, open the connection to the database.

Table creation in the database.

Insertion of data in the table



age	capital_gain	capital_loss	education_year	working_hours	occupation	sex	merital	country	race	work_class	education	relation
22	0	0	16	60	Adm-clerical	Male	Divorced	India	Amer-Indian-Eskimo	Federal-gov	10th	Husband
40	0	0	16	60	Adm-clerical	Male	Divorced	India	Amer-Indian-Eskimo	Federal-gov	10th	Husband
28	22000	0	12	40	Transport-moving	Male	Never-married	United-States	White	Private	Bachelors	Unmarried
32	22000	0	18	34	Prof-specialty	Female	Married-civ-spouse	India	Other	Federal-gov	Doctorate	Not-in-family
20	0	0	10	20	Sales	Female	Never-married	India	Amer-Indian-Eskimo	Private	Preschool	Unmarried
65	10000	0	16	40	Protective-serv	Male	Married-spouse-absent	Jamaica	Black	Self-emp-inc	Some-college	Other-relative

3.5. Export Data from Database

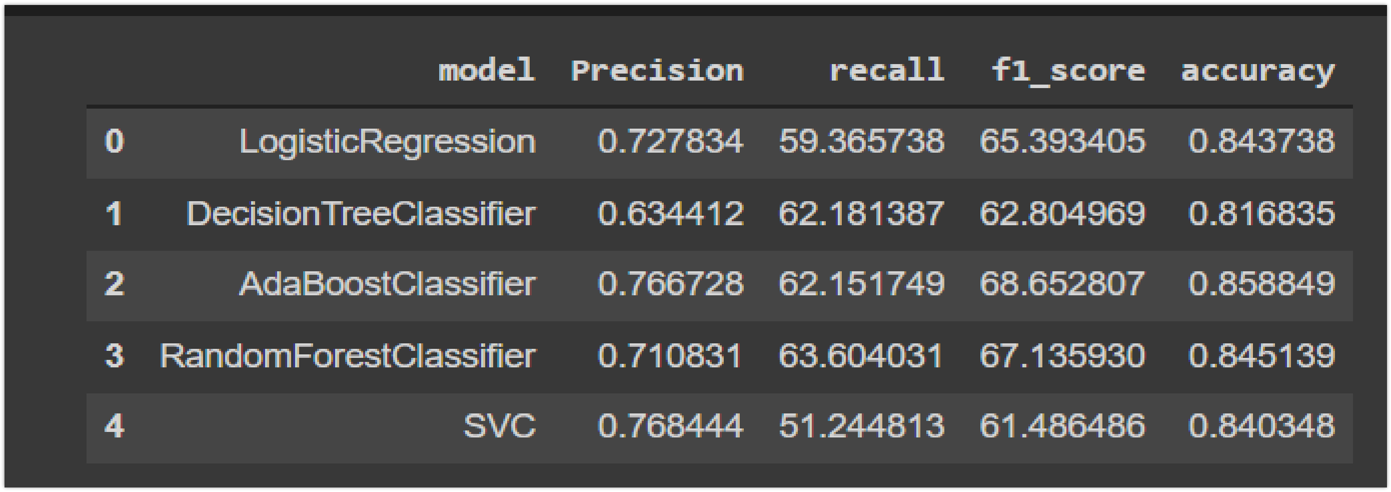
Data Export from Database - The data in a stored database to be used for Data Pre-processing and Model Training.

3.6. Data Pre-processing

Data Pre-processing steps we could use are Null value handling, Categorical to Numerical Transformation of columns , Splitting Data into Dependent and Independent Features , Remove those columns which are does not participate in model building Processes , Imbalanced data set handling, Handling columns with standard deviation zero or below a threshold, etc.

3.7 Model Creation / Model Building

After cleaning the data and completing the feature Engineering/ data Per processing. we have done splitted data in the train data and test data using method build in pre-processing file and implemented various Classification Algorithm like RandomForestClassifier and DecisionTree, AdaBoost, Svm also calculated their accuracies on test data and train data.



	model	Precision	recall	f1_score	accuracy
0	LogisticRegression	0.727834	59.365738	65.393405	0.843738
1	DecisionTreeClassifier	0.634412	62.181387	62.804969	0.816835
2	AdaBoostClassifier	0.766728	62.151749	68.652807	0.858849
3	RandomForestClassifier	0.710831	63.604031	67.135930	0.845139
4	SVC	0.768444	51.244813	61.486486	0.840348

3.8 Hyperparameter Tuning

In hyperparameter tuning we have implemented grid search cv and from that we also implemented cross validation techniques for that.

3.9 Model Dump

After comparing all accuracies and checked all ROC, AUC curve accuracy we have choose AdaBoostClassifier as our best model by their results so we have dumped this model in a pickle file format with the python module.

3.10 Data from User

Here we will collect user's requirement to predict whether a Income is more than 50K a year or not.

3.11 Data Validation

Here Data Validation will be done, given by the user.

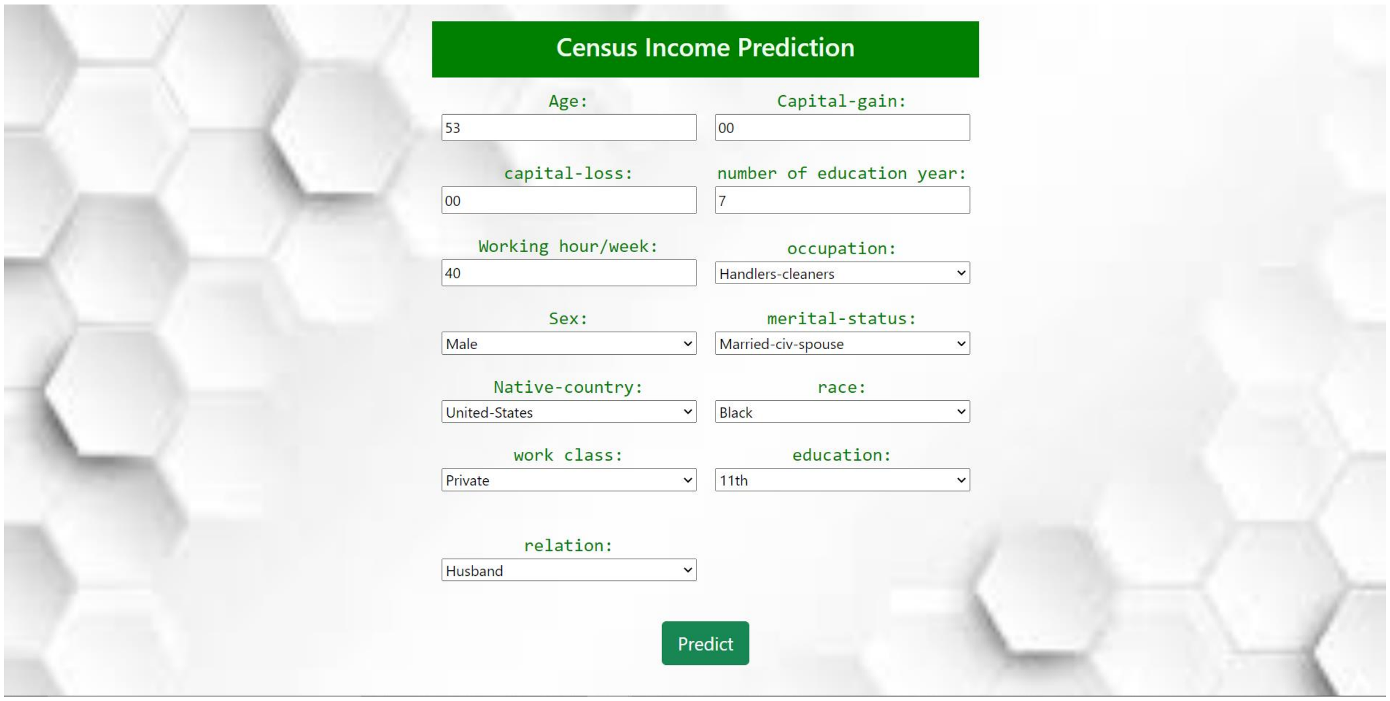
3.12 Model Call for specific input

Based on the User input will be throwing to the backend in the variable format then it converted into pandas data frame then we are loading our pickle file in the backend and predicting whether product come in backorder or not as an output and sending to our html page.

3.13 User Interface

In Frontend creation we have made a user interactive page where user can enter their input values to our application. In these frontend page we have made a form which has beautiful styling with CSS and bootstrap. These HTML user input data is transferred in variable format to backend. Made these html fully in a decoupled format.

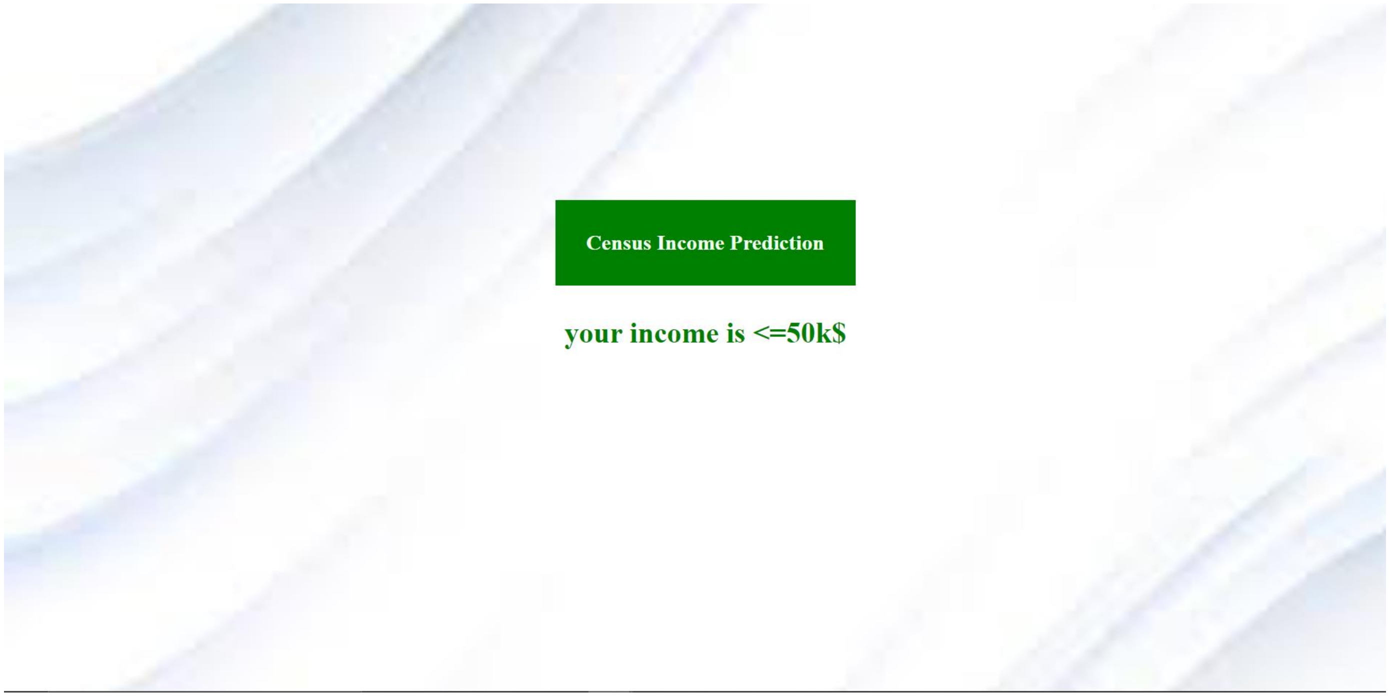
Input page :-



Census Income Prediction

Age: 53	Capital-gain: 00
capital-loss: 00	number of education year: 7
Working hour/week: 40	occupation: Handlers-cleaners
Sex: Male	merital-status: Married-civ-spouse
Native-country: United-States	race: Black
work class: Private	education: 11th
relation: Husband	

Output page :-



Census Income Prediction

your income is <=50k$

3.14 Deployment

We will be deploying the model with the help of Heroku cloud platforms.

This is a workflow diagram for the Back-Order Prediction Application ………….

4.Technology Stack

Front End	HTML, CSS, Bootstrap File
Back End	Flask, Pandas, NumPy, scikit-learn etc
Database	MySql
Deployment	Heroku

5.Directory Tree Structure

```
├── static
|   ├── images
├── templates
|   ├── welcome.html
|   ├── result.html
├── utils
|   ├── all_utils.py
|   ├── __init__.py
├── Procfile
├── README.md
├── app.py
├── Census_Income_Prediction.ipynb
├── models
├── requirements.txt
```

6.Unit Test Case

Test Case Description	**Pre – Requisite**	**Expected Resulted**
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether the User is able to Enter the data	1. Application is accessible	The User should be able to enter the data
Verify whether user is able to successfully Enter all the data.	1. Application is accessible 2. User is Enter all the data	User should be able to successfully Enter all the data.
Verify whether user is giving standard input.	1. Handled test cases at backends.	User should be able to see successfully valid results.

Verify whether user is able to edit all input fields	1. Application is accessible 2. User is logged in to the application	User should be able to edit all input fields
Verify whether the model giving the predicted output.	1.Application is accessible 2. User is able to see predicted output.	User is able to see predicted output.
Verify whether user is presented with recommended results on clicking submit	1.Application is accessible 2. user is presented with recommended results on clicking submit	User should be presented with recommended results on clicking submit