Income Prediction
Competitive Project

Submitted by : Ekata Mitra

December 20, 2022

Problem definition

A very brief overview of this data analytic case study, the US census data to build a model to predict if the income of any individual greater than or less than $50000 based on the data set. This data set was extracted by by Barry Becker from the 1994 Census database. The data set is described as below: **Target label**

$$income > 50K : binary$$

Listing of attributes

age	workclass	fnlwgt	education	education -num
marital-status	occupation	relationship	race	sex
capital-gain	capital-loss	hours-per-week	native-country	

Motivation and Objective

To start this experiment I follow few steps to implement any model:

1. **Describe the data-** The predictor attributes (independent variables features) from the Census data and the dependent variable, level of income.
2. **Acquire and Read the data-** Downloading the data directly from the source and display it.
3. **Clean the data-** Any data i.e. messy and noisy data needs to be reshaped in order to aid exploration of the data and modeling to predict target level.
4. **Explore the independent variables of the data-** A very crucial step before modeling is the exploration of the independent variables. Exploration provides great insights to an analyst on the predicting power of the variable. It helps me to understand the data or better context or clarity from his finding.

5. **Build the prediction model with the training data-** Since data like the Census data can have many weak predictors, we labelize them to speculate it clearly.

6. **Validation and Result** Here the built model is applied on validation data that the model has never seen. This is performed to determine the accuracy of the model in the field when it would be deployed.

Data PreProcessing

At the time of midterm report, I have build a relation (confusion metric) between it's numerical attributes to understand the dependencies and found 'fnlgwt' has 0 dependency. So we removed that column later. As planned in the mid-term, I applied few data Pre-processing

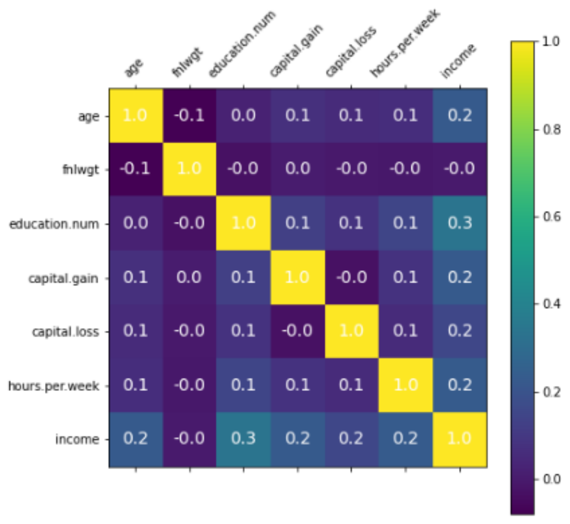


Figure 1: Confusion Metric numerical Attribute

mechanism in order to understand data in a deeper way. Here I use iterative impurity method to estimate each feature using multivariate impurer.Then I use the column transform use One-Hot encoding after segregating the columns into numeric and non-numeric columns. Here are the steps I followed for Data imputation:

- Replaced the '?' values with 'np.nan' s
- Performed 'sklearn.preprocessing.Standardscaler()' on the the numerical data
- Performed 'sklearn.preprocessing.OrdinalEncoder()' to convert the categorical data into ordinal data, the missing value was left as np.nan
- Used 'sklearn.impute.IterativeImputer' to impute the missing value iteratively. It works by performing regression with the data without any missing value and predicting the value of the missing data

- Rounded the continuous prediction
- Reverse mapped the imputed ordinal data to back to categorical data
- Converted the categorical data to one-hot data

Experiment

Since the simple data pre-processing is completed :

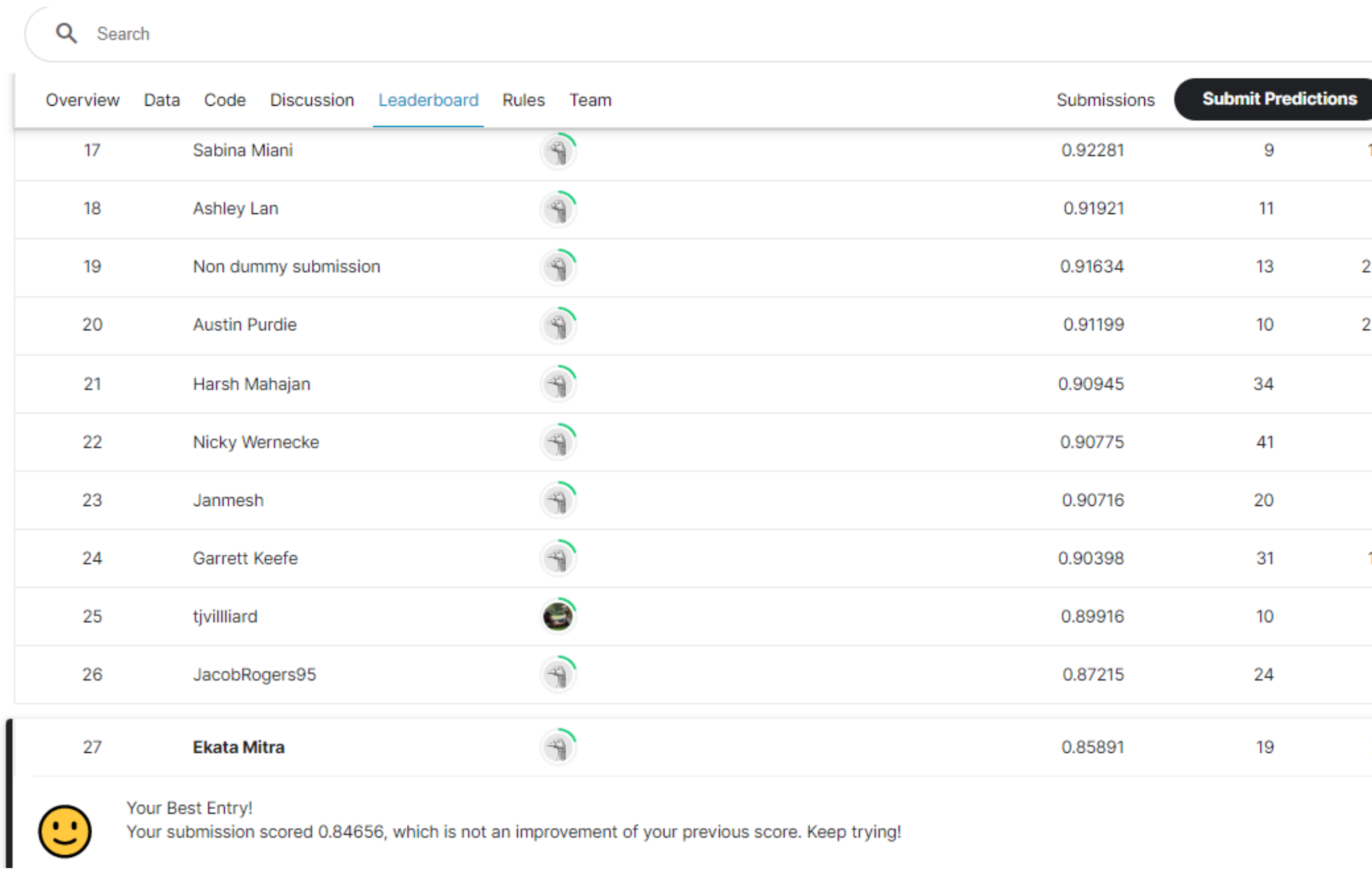
- I explored Pipeline of transforms with a final estimator.By sequentially apply a list of transforms and a final estimator with an intermediate steps of the pipeline must be 'transforms', that is, they must implement fit and transform methods.
- The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.
- For this, it enables setting parameters of the various steps using their names and the parameters, as in the example below. A step's estimator may be replaced entirely by setting the parameter with its name to another estimator, or a transformer removed by setting it to 'passthrough' .
- To start with, I explored decision tree, support vector machine, random forest, KNN,Logistic Regression and Multi layer perceptron one-by-one and implement the algorithm to classify.
- The next phase of this project, I started with Boosting algorithm like Adaboost, gradient boos, XGBoost, Extra treeclassifier algorithm with modified hyperparameters.

Experimental results

- We can speculate on the pipeline score to get the training accuracy score. Out of 6 models, I found ExtraTreeClassifier, Decision tree and MLP are very close to 87%.
- Then I did some hyperparameter tuning with tree depth with Decision tree.
- Therefore I played with hidden layers, maximum iteration , solver, activation function for MLP and found the 200 hidden layers are giving me the best performance.
- The next model ExtraTreeClassifier with few trials with hyper-parameters.

To confirm with the dataset variation in test set, I use a soft prediction by applying 'predict proba' method on test dataset where accuracy as the scoring parameter .

After this I made prediction on the test data, the score improve 92% for MLP classification and Decision Tree classifier it is showing 100%



17	Sabina Miani	0.92281	9	1
18	Ashley Lan	0.91921	11	
19	Non dummy submission	0.91634	13	2
20	Austin Purdie	0.91199	10	2
21	Harsh Mahajan	0.90945	34	
22	Nicky Wernecke	0.90775	41	
23	Janmesh	0.90716	20	
24	Garrett Keefe	0.90398	31	1
25	tjvillliard	0.89916	10	
26	JacobRogers95	0.87215	24	
27	**Ekata Mitra**	0.85891	19	

Your Best Entry!
Your submission scored 0.84656, which is not an improvement of your previous score. Keep trying!

Figure 2: snapshot from kaggle

However when we submitted data into Kaggle with their own estimation with csv file,I found:

Accuracy on the test dataset is now **85.9%** in comparison to **78.16 %** with **MLP classifier**.

Github Link

https://www.kaggle.com/code/ekatamitra/cs-6350-ml-project

Future Plan

I will work on the pre-processing, preferably again try different classifier naive bayes with modified parameters.