Machine Learning Engineer Nanodegree

Capstone Project

September 28th, 2021

I. Definition

Project Overview

Information about an individual's income level is valuable in many areas and domains, e.g., it would be very useful for a bank to know a customer's income level while determining his/her segment. Another example can be marketing campaigns; if you know the income levels of your target customers, you can offer them the products which they can afford. As a result the success of the marketing campaign would increase.

But most of the time this information is confidential and not easily accessible. Although we cannot easily have the knowledge about the incomes of individuals, we can at least try to estimate an approximate amount using the other information we have. This project will be about predicting the income levels of people.

Datasets and Inputs:

Census Income Data Set from UCI Machine Learning Repository [1] is used in the project. Original dataset can be found here [2].

Data was originally extracted from the Census Bureau database. Extraction was done by Barry Becker from the 1994 Census database. The dataset was split into train and test sets by the creator. Data contains 14 attributes and the target label ('>50K' and '<=50K').

2 separate data files are used from the UCI Machine Learning Repository:

- adult.data: train data
- adult.test: test data

Data files contains both independent (attributes) and dependent (the target label) variables.

It can be said that the dataset is imbalanced as the probability for the label '>50K' is 23.93% and the probability for the label '<=50K' is 76.07%.

Problem Statement

The goal of the project is predicting whether a given adult makes more than $50,000 a year. Predictions will be based on a certain set of demographic features of an individual. Namely this is a binary classification problem where a person with an income more than $50K per year is labeled as 1 and 0 otherwise.

Since we have the target labels (the output/dependent variable) this is a supervised learning task and as we stated before this is a binary classification problem. As it is needed in any machine learning problem, we need to explore (exploratory data analysis) and preprocess (feature engineering) out data. Also the imbalance in the dataset should be taken into account while developing the models and determining the evaluation metrics.

Metrics

Because of the imbalance, using only accuracy for evaluation of the model would not be sufficient as the model would be prone to make false negative predictions. False negative predictions would cause a low recall value.

To be able to keep both false positives and false negatives under control and to evaluate the model, we can use ROC - Receiver Operating Characteristic - Curve and AUC - Area Under the Curve - Score (Precision and recall can be used too, but AUC combines these metrics, however, these metrics are also observed).

II. Analysis

Data Exploration & Exploratory Visualization

Train and test sets were in different files. I have loaded these data files separately and then combined train and test data sets into a single pandas DataFrame in order to make things simpler during exploratory data analysis – EDA – step.

INCOME is the target/output variable, and it is not in a binary format. Another problem with this variable is that there is an inconsistency between train and test datasets. In train set it is labeled as <=50K and >50K while it is labeled as <=50K. and >50K. in test set. To facilitate the EDA process this problem is fixed at initially by making them consistent.

Important observations about the dataset are summarized below:

- Training and test datasets have 48842 data points with 15 columns at total (including target column - income). The test set consists of 16281 records, while the train set consists of 32561.

- Information about variables:
  AGE: continuous.
  WORKCLASS: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
  FNLWGT: continuous.
  EDUCATION: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
  EDUCATION-NUM: continuous.
  MARITAL-STATUS: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
  OCCUPATION: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
  RELATIONSHIP: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
  RACE: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
  SEX: Female, Male.
  CAPITAL-GAIN: continuous.
  CAPITAL-LOSS: continuous.
  HOURS-PER-WEEK: continuous.
  NATIVE-COUNTRY: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
  INCOME: <=50K, >50K

- **Imbalanced dataset:** Target variable labeled 1 (income more than $50K): 24,08% of the dataset. This may result in overfitting towards data labeled as 0, which accounts for some false negatives; cases in which data labeled (1) is incorrectly predicted as labeled (0). Even if a model labels all of the data as 0, it will still have 76% accuracy. We must take this issue to account while developing the model and choosing the performance evaluation metrics.

- **Number of categorical variables:** 9

	WORKCLASS	EDUCATION	MARITAL-STATUS	OCCUPATION	RELATIONSHIP	RACE	SEX	NATIVE-COUNTRY	INCOME
0	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	United-States	<=50K
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	<=50K
2	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States	<=50K
3	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States	<=50K
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba	<=50K

- **Missing values** are encoded with a '?', 3 categorical variables have missing values. Percentage of missing values are not high.

```
WORKCLASS          0.057307
OCCUPATION         0.057512
NATIVE-COUNTRY     0.017546
```

- Most of the categorical variables show **low cardinality**.

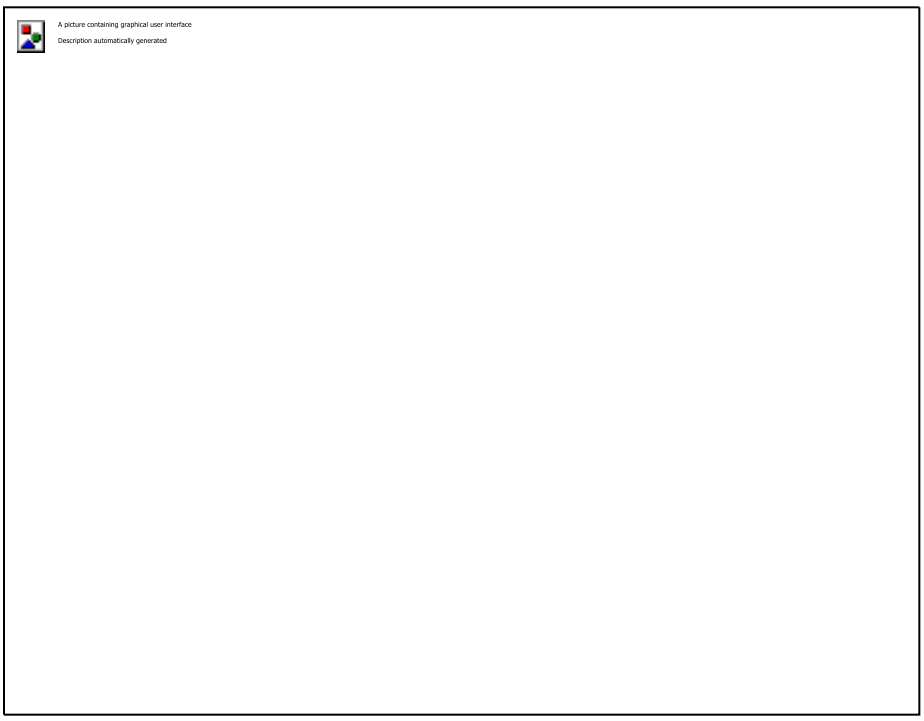
- **Rare Labels**: Some categorical variables show multiple labels that are present in less than 1% of the houses.

- **Number of numerical variables:** 6 (including output variable - income)

	AGE	FNLWGT	EDUCATION-NUM	CAPITAL-GAIN	CAPITAL-LOSS	HOURS-PER-WEEK
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40

- Numerical **EDUCATION_NUM** and categorical **EDUCATION** variables keep the same data in different formats. EDUCATION is an ordinal variable, has categories with a particular order associated with it. EDUCATION_NUM already keeps the ordinal encoded data; categorical EDUCATION column can be dropped.

- **Correlation**: Correlation Matrix shows that there isn't a very high linear correlation between any of numerical variables and the target variable.
  FNLWGT variable shows a negative correlation close to zero with target variable, can be dropped.



- Numerical variables **are not normally distributed**.

Algorithms and Techniques

I used tree-based classification algorithms for predicting the income level. Following classifiers are used to develop machine learning models:

1. **Random Forest:** Decision tree algorithms suffer from high variance; that is, if we randomly split the training data into 2 parts, and fit decision trees on both parts, the results could be quite different. To overcome this disadvantage of decision trees we can use bagging (bootstrap aggregating). Bagging is based on two things: averaging (reduces variance) and bootstrapping (plenty of training datasets).
   Random Forest is a very efficient statistical learning method which builds on the idea of bagging, but it provides an improvement by decorrelating the trees. This reduces the variance when we average the trees.

2. **Adaptive Boosting - AdaBoost:** Adaptive Boosting is an ensemble method. The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator [3].
   Adaptive Boosting is formulated by Yoav Freund and Robert Schapire. It can be used in conjunction with many other types of learning algorithms to improve performance. The individual learners can be weak, but as long as the

performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner [4].

3. **Gradient Boosting:** Gradient Tree Boosting is a generalization of boosting to arbitrary differentiable loss functions. It can be used for both regression and classification problems. It builds the model in a sequential way.
   AdaBoost and related algorithms were recast in a statistical framework first by Breiman calling them ARCing algorithms. This framework was further developed by Friedman and called Gradient Boosting Machines. Later called just gradient boosting or gradient tree boosting.

Besides these I used **Synthetic Minority Over-sampling Technique – SMOTE** in order to deal with the imbalanced data. This technique deals with imbalanced datasets by oversampling the minority class.

Benchmark

Census Income Data Set is an old and known dataset. Previous work and results can be found about it. It is also available in kaggle [6]. Chakrabarty et. al. [7] obtained 88.16% test set accuracy and 0.93 AUC score using Gradient Boosting Classifier. A literature review about predicting income levels and some other benchmark model results can also be found in this paper.

III. Methodology

Data Preprocessing

After exploring data, it can be preprocessed and be made ready for modelling. But before beginning to process the data, it is important to separate the data into train and test sets. When we engineer features, some techniques learn parameters from data and this may cause data leakage from test set to train set. It is important to learn parameters only from the train set in order to avoid over-fitting.

Since I already had separate DataFrames for train and test sets, I used them the feature engineering step.

Data preprocessing steps are listed below:

- Unnecessary columns are **dropped**: EDUCATION, FNLWGHT
- **Missing values:** To prevent data loss and not to bias the predictions in favor of most occurred category, missing values are replaced with a new category "Missing".

- **Rare Labels:** Labels that are under-represented in the dataset tend to cause over-fitting. Variables that are present in less than 1% of the observations, are replaced by the string "Rare" and reduced to a new category.

- **Encoding Categorical Variables:** One-hot encoding method is used for transforming categorical variables into numerical variables. scikit-learn's OneHotEncoder encoder is used, as it can handle test data with new labels not found in the training data.

- **Transformation of Numerical Variables:** Since the numerical variables are not normally distributed, they are log transformed.

- **Scaling of Numerical Variables:** The scale of numerical columns are standardized in order to consistently compare the values of different features. MinMaxScaler is used for scaling.

- **Imbalanced Data:** One approach to deal with imbalanced datasets is oversampling the minority class. **Synthetic Minority Over-sampling Technique – SMOTE** is used for this purpose.

Implementation

3 different classifiers are initialized:

- Random Forest

- Adaptive Boosting

- Gradient Boosting

These models are trained mostly using their default parameters. Only the `n_estimators` parameter is set to 500. Also, the `random_state` parameter is set to 0 for reproducibility. Initialized models are as below:

```
rfc = RandomForestClassifier(n_estimators=500, random_state=0)
abc = AdaBoostClassifier(n_estimators=500, random_state=0)
gbc = GradientBoostingClassifier(n_estimators=500,
random_state=0)
```

Implementation was smooth. I did not encounter any complications or difficulties during coding process.

Gradient Boosting Classifier performed the best both for accuracy and AUC score. Also, it has relatively balanced precision and recall values which results with the highest F1-Score, that is the harmonic mean of these two metrics.

Random Forest performed worst, it also overfitted to training data.

Base Results		
Model	**Test Accuracy**	**Test AUC Score**
RANDOMFOREST	0.8372	0.8895
ADAPTIVEBOOSTING	0.8477	0.9154
GRADIENTBOOSTING	0.8585	0.9215

Refinement

Gradient Boosting Classifier was the best performing among the 3 algorithms and its performance is tried to be improved using hyperparameter tuning with grid search. Before grid search, I tried coarse-to-fine search: finding the regions of high potential, then using a fine grid search in these regions to find best setting of the hyperparameters. But I was not able to improve the base result.

Then I tried grid search directly and tuned only 3 hyperparameters using the following values:

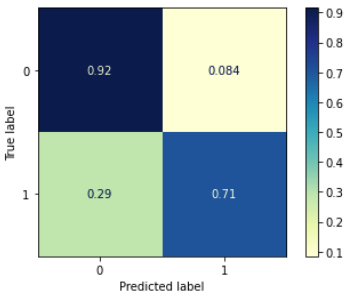
```
'n_estimators':[200, 500, 700],
'learning_rate':[0.2, 0.15, 0.1, 0.05],
'min_samples_split' : np.arange(2, 6, 2),
```

Surprisingly, after fitting only 2 folds for each of 24 candidates, totaling 48 fits, both test accuracy and AUC scores improved as below:

```
Fitting 2 folds for each of 24 candidates, totalling 48 fits

Accuracy: 0.8673
AUC Score: 0.9242

('precision', 0.722942577401429) ('recall', 0.7103484139365575) ('f1-score', 0.7165901639344262) ('support', 3846)
```



IV. Results

Model Evaluation and Validation

As it is stated before initially 3 different tree-based models were used for classification. Then the best performing one was tuned. After hyperparameter tuning the final Gradient Boosting Classifier model is obtained with the following set of parameters:

```
'ccp_alpha': 0.0,
'criterion': 'friedman_mse',
'init': None,
'learning_rate': 0.15,
'loss': 'deviance',
'max_depth': 3,
'max_features': None,
'max_leaf_nodes': None,
'min_impurity_decrease': 0.0,
'min_impurity_split': None,
'min_samples_leaf': 1,
'min_samples_split': 4,
'min_weight_fraction_leaf': 0.0,
'n_estimators': 700,
'n_iter_no_change': None,
'random_state': 0,
'subsample': 1.0,
'tol': 0.0001,
'validation_fraction': 0.1,
'verbose': 0,
'warm_start': False}}
```

Justification

Benchmark model has 88.16% test set accuracy and 0.93 AUC score. My tuned model has 86.73% and 0.9242 AUC Score on test set. Although it is not as good as the benchmark model, it can be said that it performs close to it.

V. Conclusion

It can be said that if we have the right data, it is possible to predict the income levels of individuals with a relatively high accuracy. Some further modifications and use of other techniques like Principal Component Analysis – PCA – for dimensionality reduction can help to improve the results. Another field of improvement may be applying clustering before classification and using this clustering data as a feature.

References

[1]. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2]. https://archive.ics.uci.edu/ml/datasets/census+income

[3]. https://scikit-learn.org/stable/modules/ensemble.html#ensemble

[4]. https://en.wikipedia.org/wiki/AdaBoost

[5]. https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

[6]. https://www.kaggle.com/uciml/adult-census-income

[7]. Chakrabarty, N., & Biswas, S. (2018). A statistical approach to adult census income level prediction. 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN). https://arxiv.org/pdf/1810.10076v1.pdf