CSE 708: Application of Data analytics and Engineering
Class Project: Census Income Binary Classification

Robert Akinie

December 10, 2023

Project Scope

The scope of this project is on an end-to-end predictive modeling machine learning problem, with the application on binary classification. The focus of this project would be on the Census Income dataset, extracted from the US Adult Census bureau database, and the primary problem is a prediction task: determining whether a person from the Census database made over $50, 000 a year.

The project can be summarized into two tasks: prediction task being performed utilizing various classifiers and other machine learning (ML) methods, including those explored in the class, as well as exploring the impact of sampling on model performance, both training and in deployment.

Dataset Overview and Descriptive Statistics

This section discusses properties associated with the data. The Census Income dataset was extracted from the 1994 US Adult Census bureau database [1] , with the task of predicting whether a given adult made more than $50,000 a year, based on the following attributes from the Census

Age: the age of an individual
Workclass: employment status
Fnlwgt: census sampling
Education: level of education
Marital status: marital status of an individual
Occupation: general type of occupation
Relationship: represents what this individual is relative to others
Race: descriptions of an individual's race
Sex: biological sex of the individual
Capital gain/loss: capital gains for an individual
Hours per week: hours an individual has reported to work per week
Native country: country of origin for an individual
Income: whether or not an individual makes more than $50,000 annually

The dataset has 32561 entries, with both categorical and numerical attributes, and has the following distribution: 24% entries labeled with >50k and 76% entries labeled with <=50k, before cleaning the data, which shows how highly unbalanced this data is. This dataset was specifically chosen in order to fulfill the second task in the project scope section. The dataset contained some unknown values encoded as '?', even though there were no missing values. Those values were then encoded as NaN's and removed. THe dataset was mixed with both numerical and categorical variables.

Following the data cleaning, graphs were generated to display income distributions for some of the attributes. The purpose was to learn some insights about the most/least helpful features that contribute to the prediction task. The graphs can be found in the Appendix section, which is the python file that contains code fro the project.

Feature Engineering

For this project, data cleaning before and after descriptive statistics, train/validation splits, feature encoding and scaling were the feature engineering straps taken. The dataset was split into training and validation sets while maintaining the distribution across both splits. This was done to maintain an equal ratio of target class types, and remove any bias that may have otherwise been introduced.

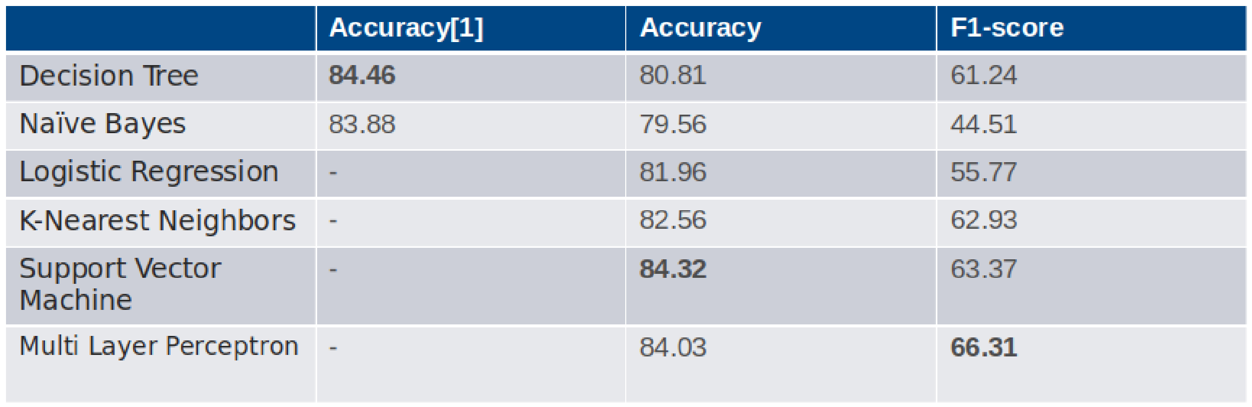
Following the split, the categorical data was encoded using label encoder. The issue with label encoder is that it can introduce a level of hierarchy within the variable values, which may skew some machine learning models. Nonetheless, this was chosen over the Pandas encoder “get_dummies(),” for constraints, as well as displaying a correlation matrix. No features were taken out, in order to determine the baseline performance, and then for model deployment, features that contributed the least were manually taken out.

Model Preparation and Algorithm Evaluation

This step was about working with a number of machine learning algorithms that are good at exploiting the structure of the dataset. For this project, these were the ML models and algorithms evaluated on this dataset: Logistic Regression, K-Nearest Neighbors, Decision Trees, Naïve Bayes, Support Vector Machine and Multi Layer Perceptron. PCA was also performed, but there was marginal improvement in baseline performance, and so was not considered.

The data was then trained on these models, with 10-fold cross validation, keeping the same distribution within each fold. Performance metrics used were Accuracy and F1-score. THe following table shows the baseline performance of these models. The **bold** values represent the best performing model for each metric.

Table 1. Model performance



	Accuracy[1]	Accuracy	F1-score
Decision Tree	**84.46**	80.81	61.24
Naïve Bayes	83.88	79.56	44.51
Logistic Regression	-	81.96	55.77
K-Nearest Neighbors	-	82.56	62.93
Support Vector Machine	-	**84.32**	63.37
Multi Layer Perceptron	-	84.03	**66.31**

Performance Improvement

We investigated various ways of improving the performance of the model, such as ensemble methods. We looked at the performance of algorithms on this problem by using ensemble methods, and evaluated four different ensemble machine learning algorithms, two boosting and two bagging methods. The table below displays the performance metrics scores with the ensemble methods.

Table 2. Model Performance Improvement: Ensembling

	Accuracy	F1-score
AdaBoost	85.55	67.29
Gradient Boosting	**86.00**	**68.14**
Random Forest	84.54	67.26
ExtraTree	83.86	65.08

Oversampling

The imbalanced classification problem is what arises when there is a severe skew in the class distribution of the training data. One way the imbalanceness may affect the model is when the selected algorithms completely ignores the minority class. The reason this is an issue is because in this instance, the minority class, which is income greater than 50% is the class that we are most interested in.

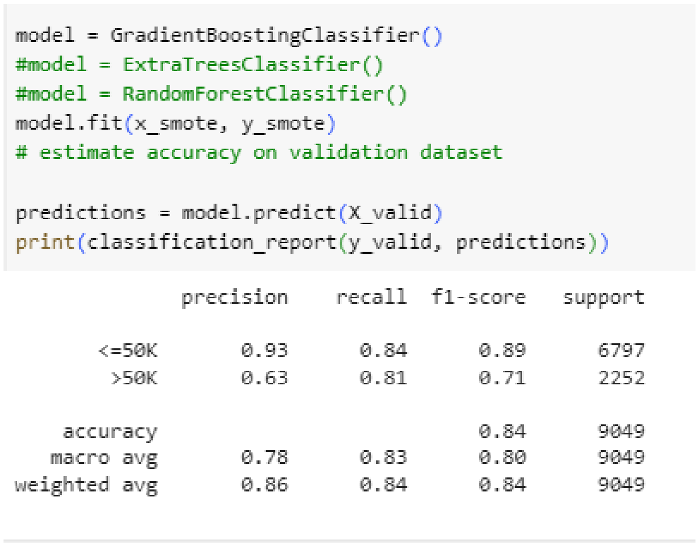
Given that the dataset is a very imbalanced dataset, considering sampling is a good technique in mitigating the problem of performance metric bias to the majority class. As a result, oversampling was done with SMOTE() to increase the minority class. Table 3 displays the performance metric comparison.

Table 3. Model performance improvement: Oversampling

	Accuracy		F1-score	
	Imbalance Set	Oversamp Set	Imbalance Set	Oversamp Set
Decision Tree	80.81	85.12	61.24	85.05
NaïveBayes	79.56	69.73	44.51	60.67
Logistic Regression	81.96	76.74	55.77	76.61
KNN	82.56	84.81	62.93	85.67
SVM	84.32	82.76	63.37	83.71
MLP	84.33	82.90	65.59	83.60
AdaBoost	85.55	84.57	67.29	84.89
GMB	**86.00**	86.01	**68.14**	86.35
RF	84.63	89.00	67.10	88.97
ExtraTree	83.69	**89.38**	65.16	**89.40**

Final Model Performance

In this section we finalized the model by training it on the entire training dataset and making predictions for the hold-out validation dataset to confirm our findings. For the model, we picked the Gradient Boosting Classifier, which is an ensemble classifier.



```
model = GradientBoostingClassifier()
#model = ExtraTreesClassifier()
#model = RandomForestClassifier()
model.fit(x_smote, y_smote)
# estimate accuracy on validation dataset

predictions = model.predict(X_valid)
print(classification_report(y_valid, predictions))
```

```
              precision    recall  f1-score   support

       <=50K       0.93      0.84      0.89      6797
        >50K       0.63      0.81      0.71      2252

    accuracy                           0.84      9049
   macro avg       0.78      0.83      0.80      9049
weighted avg       0.86      0.84      0.84      9049
```

Figure 1. Final Model Performance

Given that out validation accuracy reduced to 71%, the most likely reason would be that there was overfitting of the model to the training data, and poor generalization to new data in deployment. We can then see the impact of oversampling on the training versus validation splits, since it contributed to the model overfit.

Conclusion

In this project, we set out to perform the prediction task of determining whether a person, makes over $50, 000 a year, based on the provided data. We also looked at the importance of sampling on model performance. Possible Improvements in this project include getting more real data, hyper parameter tuning, as well as considering better performance algorithms.

References

[1] Kohavi, Ron. "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." Kdd. Vol. 96. 1996.
[2] Brownlee, Jason. "Machine learning mastery with python understand your data." Create Accurate Models and Work Projects End-To-End 285.118 (2016): 35