1 American Census

1.1.1 (1 point) Explore the features and target variables of the dataset. What is the right performance metric to use for this dataset? Clearly explain which performance metric you choose and why. Use no more than 125 words.

We prefer to choose F1-score as our performance metric for this dataset. There are mainly three reasons for this choice.

- This is a real-life binary classification problem, for which F1-score is widely used.
- When we explore the label set, we find that the set is imbalanced. There are approximately 4K people have the salaries above the threshold, but about 12K people are below that threshold. F1-score is a better metric to evaluate the imbalanced class distribution which has a large number of actual negatives, because the return value of F1-score is a harmonic mean of precision and recall.
- We are familiar with this type of metric.

1.1.2 (1 point) Algorithmic bias can be a real problem in Machine Learning. So based on this, should we use the Race and the Sex features in our machine learning algorithm? Clearly explain what you believe, also provide us with arguments why. Note this question will be graded based only on your argumentation. Use no more than 75 words.

It depends on the target or the goal of our model. If we advocate using these features to dig out and to report the problem relative to racial and gender discriminations, or even age and academic discriminations, these features should be taken into consideration, since we want to use these features to find out the biases, to report them, and even to reduce and eliminate them, in order to build an impartial and harmonious society.

1.2.1 (2 points) This dataset hasn't been cleaned, yet. Do this by finding all the missing values and handling them. How did you handle these missing values? Clearly explain which values were missing and how you handled them. Use no more than 100 words.

There are 1573 samples, which contain at least one missing feature values, in the training set. We used two different ways - dropping the incomplete samples or filling the missing value - to preprocess the data. Dropping them is the simplest method, but we will lose about 10% training samples; whereas, filling them will manually add noise. For the filling method, we will create a new group " Unknown" for the missing categorical features and use mean value to fill the missing numeric features.

1.2.2 (2 points) All Scikit-learn’s implementations of these algorithms expect numerical features. Check for all features if they are in numerical format. If not, transform these features into numerical ones. Clearly explain which features you transformed, how you transformed them and why these transformations. Use no more than 75 words. (You might want to read the preprocessing documentation of Scikit-learn for handy tips.)

There are some categorical features which are "workclass", "education", "marital-status", "occupation", relationship", "race", "sex" and "native-country". We used OrdinalEncoder to encode these features. OrdinalEncoder can change the categorical features into numeric integer formats, which will become acceptable to the implementations.

1.2.3 (Bonus 2 point) Have you done any other data preprocessing steps? If you did, explain what you did and why you did it. Use no more than 100 words.

We have done two more preprocessing steps.

1. In the feature of education, we find that the division is too specific for primary educations compared to other periods of educations. Therefore, we merge some of the primary educational periods according to the US educational system, since over-specialized data may lead to overfitting.

2. After encoding the features, the scopes of them vary with each other. Therefore, we created two re-scaled versions for the data to eliminate the varying scales. The first one is a normalized version, and another one is a standardized version.

1.3.1 (1 point) Now set up your experiment. Clearly explain how you divided the data and how you ensured that your measurements are valid. Use no more than 100 words.

We decide to use KFlod cross-validation to train and test our models because it can make full use of the dataset, and avoid the overfitting problem to some extent. The value of **n_splits** is set to 10, which means it will use 1 fold for validating, the rest 9 folds for training in one round. In terms of ensuring our measurements, we plotted the learning curve for each training process in order to compare the F1-scores.

1.3.2 (2 points) Fit the five algorithms using the default hyper-parameters from section 2.1. Create a useful plot that shows the performances of the algorithms. Clearly explain what this plot tells us about the performances of the algorithms. Also, clearly explain why you think some algorithms perform better than others. Use no more than 150 words and two plots (but 1 is sufficient).

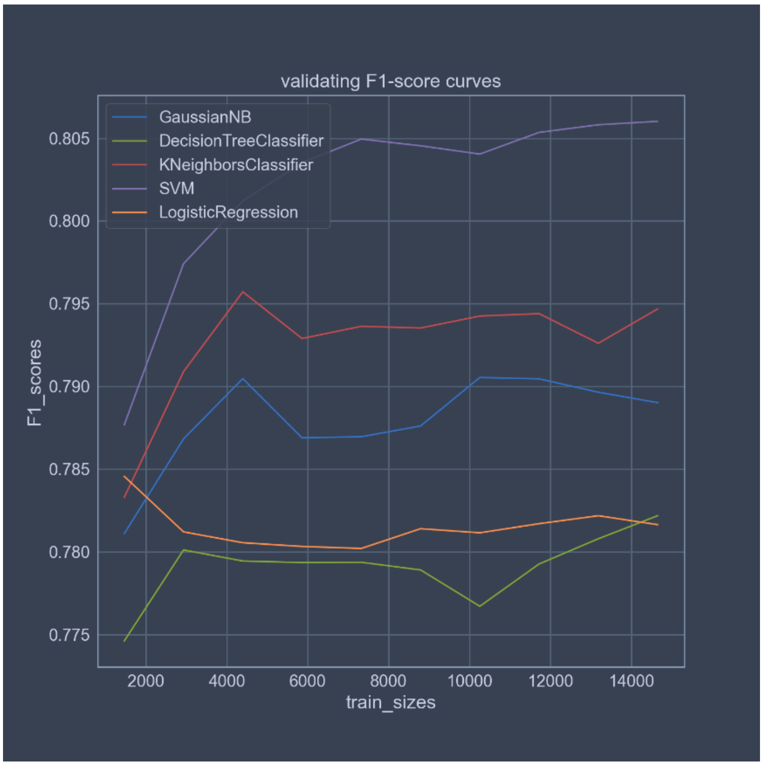
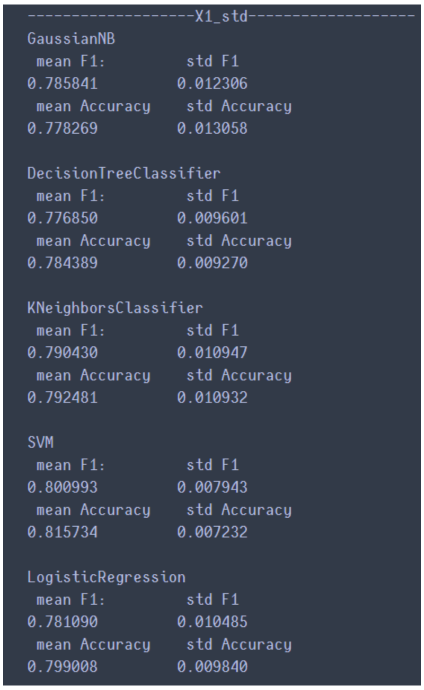


Figure 1 Testing Learning Curve



```
------------------X1_std------------------
GaussianNB
 mean F1:         std F1
0.785841        0.012306
 mean Accuracy    std Accuracy
0.778269        0.013058

DecisionTreeClassifier
 mean F1:         std F1
0.776850        0.009601
 mean Accuracy    std Accuracy
0.784389        0.009270

KNeighborsClassifier
 mean F1:         std F1
0.790430        0.010947
 mean Accuracy    std Accuracy
0.792481        0.010932

SVM
 mean F1:         std F1
0.800993        0.007943
 mean Accuracy    std Accuracy
0.815734        0.007232

LogisticRegression
 mean F1:         std F1
0.781090        0.010485
 mean Accuracy    std Accuracy
0.799008        0.009840
```

Figure 2 F1-Score/Accuracy Comparison

In figure 1, it shows the learning curve of testing curves. Using the default setting, SVM, always owning the highest F1-score no matter how many samples in the set, is the best algorithm among others.

The performances of the algorithms strongly relative to the training samples. If SVM performances well on this dataset, this dataset probably has a relatively clear margin of binary classification, which means there are a few overlapping samples of two different classes. Following that, KNN has second-highest accuracy. It considers every sample, so its tolerance may be lower. The third one is Gaussian classifier, which strongly relies on the distribution of the samples. After that is Decision Tree, which is weaker for predicting continuous feature values. The final one is LR, a kind of statistic algorithm and unlike SVM which is a kind of geometric algorithm.

1.3.3 (2 points) Now perform hyper-parameter tuning on the key hyper-parameters you have previously identified. Clearly explain what you did to be systematic, what you did to get fair results, what trade-off accuracy vs resources trade-off, etc. Use no more than 200 words. Note: First focus on tuning the default hyper-parameters, this should be sufficient. Only look at others if time permits it.

Systematic: In order to exhaustively search the best parameters, we adapt

GridSearchCV. Using this method, we can define a group of values for each parameter of a model, which will be tested independently each time, and GridSearchCV will return the best combination of values for each algorithm.

Fairness: We cannot access to the best values before training and validating. Therefore, we need to trial. Initially, we will set a rough range of values to test their performances. Then, we will narrow down the ranges of the parameters according to the feedback of GridSearchCV. We also set multiple random states so as to get the best result for each model, because the random state can influence the accuracy, even the effect is not significant.

Trade-off: The first limitation is the number of samples, and this is also the reason why we do not want to drop any samples during preprocessing. Therefore, every round we will feed the model with all samples. The second limitation is the wide range of parameters, and especially it will spend a long period for training if they are set too high. Therefore, we need to stop narrow down when the accuracy is not significantly promoted after tuning.

1.3.4 (2 points) Compare the performance of the algorithms with and without hyper-parameter tuning. How did the tuning affect your result? Clearly explain the results and the differences. Use no more than 100 words and two plots (but 1 is sufficient).

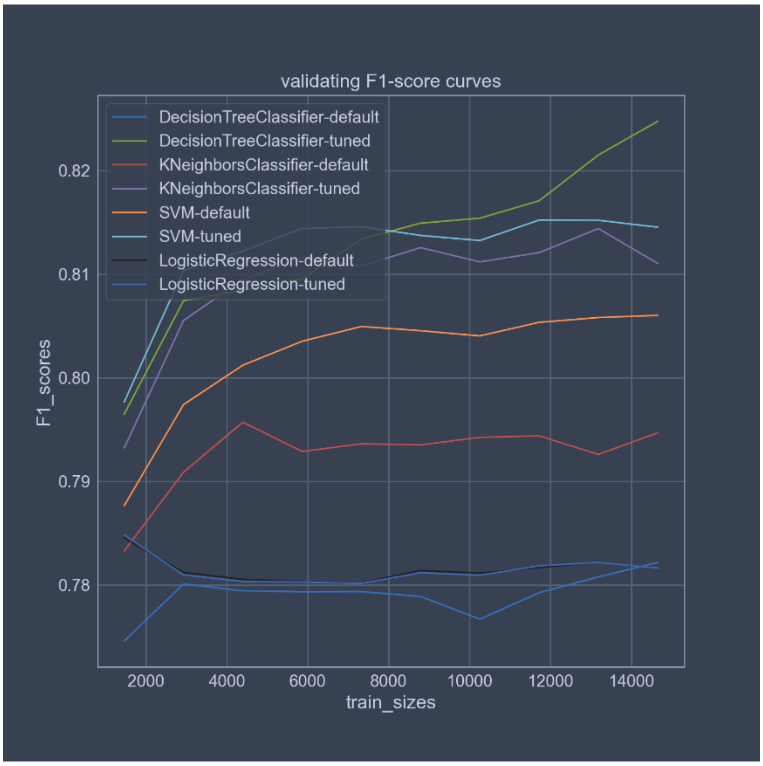


Figure 3 Hyper-parameter Tuning

After hyper-parameter tuning, all the accuracies of the algorithms have been improved significantly excluding Logistic Regression. Decision Tree Classifier, which was the worst algorithm, now becomes the best one. SVM still performs well and locates at the second positions. Following that are K-Neighbors and Logistic Regression classifiers. We do not provide the Gaussian here, since it is not required to adjust hyper-parameters. Back to the current best model, DecisionTreeClassifier, it now has the trend of ascending under the increasing of training size. Therefore, if there were more training samples, its accuracy would probably keep climbing up.

1.3.5 Select your best algorithm for this dataset and use it to make your predictions for the unknown samples. Please note in your algorithm which algorithm you chose.

We decide to choose Decision Tree Classifier with the following hyper-parameters:

- max_depth=8,
- min_samples_leaf=4,
- random_state=40