Comparative Analysis of various Machine Learning Algorithms on US Adult Income Dataset.

ABSTRACT: The work deals with applying the complete Machine learning lifecycle on the Adult Income Census dataset for predicting the income range of an individual. Main objective lies in applying the data pre-processing steps for preparation of this dataset and exploring the various machine learning algorithms for building and identifying the best performing models that can undergo the hyper-parameter optimization. Best model proved to be Gradient Boosting classifier with the weighted average of 84.00% for f1 scores of both minority and majority classes after tuning its hyper-parameters. In this project, feature selection is taken as the research element where I have integrated various feature selection mechanisms and evaluated the impact of integrating them into my machine learning pipeline.

1. INTRODUCTION

Unequal distribution of wealth in any of the nation is the problem under consideration and will lead to economic inequality. Various governments have been trying to find the solution to equalize the money across the population that will help in achieving the economic stability. Machine learning has the ability to deal with this problem by analysing the key factors that are necessary to improve the income of a person. The predictions for the income range, by our algorithm by finding the patterns in the data can help in making the appropriate action plans and strategies to improve the economy of a country.

The dataset I have chosen for this research project is US Census income dataset from the UCI machine learning repository collected in 1994 by Barry Becker. This is a classification problem where the aim is to predict whether the income range of an individual is greater than or less than 50K dollars, given the various demographic, social and economic features. Dataset contains the records of the 48842 persons. There are a total of 14 features out of which 8 are categorical features and 6 are continuous. The target feature is the income which is a binary column containing 2 classes (>50K or <=50K) that needs to be predicted by our machine learning model.

Some of the relevant work associated with this adult census dataset has been done in past. In [1], Lazar predicted the income range using support vector machine and also used PCA for the feature selection because of generation of many features due to one hot encoding. Deepajothi in [2] performed a comparative analysis of various classification algorithms like Decision tree, Bayesian networks, Random forest and Naïve Bayes. Bekena [3] implemented a Random Forest classifier for this problem and achieved an accuracy of 85% on the test set.

This paper has been divided into six sections and starts with Abstract, Introduction, Research Methodology, Evaluation and ends with Conclusion & Future Work.

2. RESEARCH

I incorporated various feature selection mechanisms into my methodology and results. Technically, after applying each of the feature selection technique I should perform all the steps again from obtaining the best models to the tuning of their parameters and finally evaluating the performance on the test set. As the dataset is big enough and even increased after applying SMOTE (taking much time because of parameter tuning) therefore I chose to evaluate the impact of these feature selection techniques using the obtained best hyper-parameters of the best model (Random Forest and Gradient Boosting) on the test set using various evaluation metrics. Then I carried out the comparison of the performance on the test set with and without feature selection with these tuned models.

Feature Selection is the process of ranking or quantifying the contribution of each feature in a dataset on the basis of the relationship of that feature column with the target feature column.

Some of the feature selection mechanisms I applied are as follows

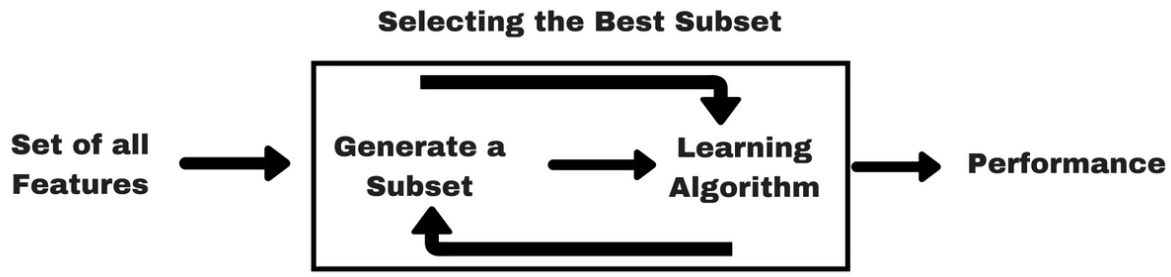
1) Univariate Feature Selection - In Univariate feature selection statistical tests can be used to select those features that have the strongest relationship with the output variable. We have make use of SelectKBest class of Scikit learn along with the chi-squared statistical test for non-negative features to select the top 10 features out of 13 available features for our census dataset. There is the score associated with each of the features meaning that higher the score is, the stronger the relationship is with target feature. We removed the 3 features with the lowest scores that were race, native country and occupation.

2) Tree Based Feature Selection (Feature Importance) – Feature importance is the property of the tree based models like random forest or decision tree. They work by reducing the level of uncertainty or impurity in the data by making use of metrics such as gini or entropy. This mechanism for the feature selection builds many trees and calculates the average reduction in uncertainty achieved by each of the individual feature across all trees and uses this as means of feature ranking.

3) Feature Selection using Correlation Heat-map-

We plotted a correlation matrix on the basis of *Pearson correlation coefficient*. This matrix is the representation of the correlation of each feature with every other feature in the dataset. The value of this coefficient is between -1 to 1. If the value of this coefficient between 2 features approaches to -1 or negative then those features are negatively correlated meaning that the increase in one feature decreases the value of other feature. If the value of coefficient is positive or approaches to +1 then these features are positively correlated meaning that increase in one feature increases the value of other feature. And if this approaches to 0 then there is no correlation between the features.

4) Greedy Based feature selection (Recursive Feature Elimination with cross validation)-

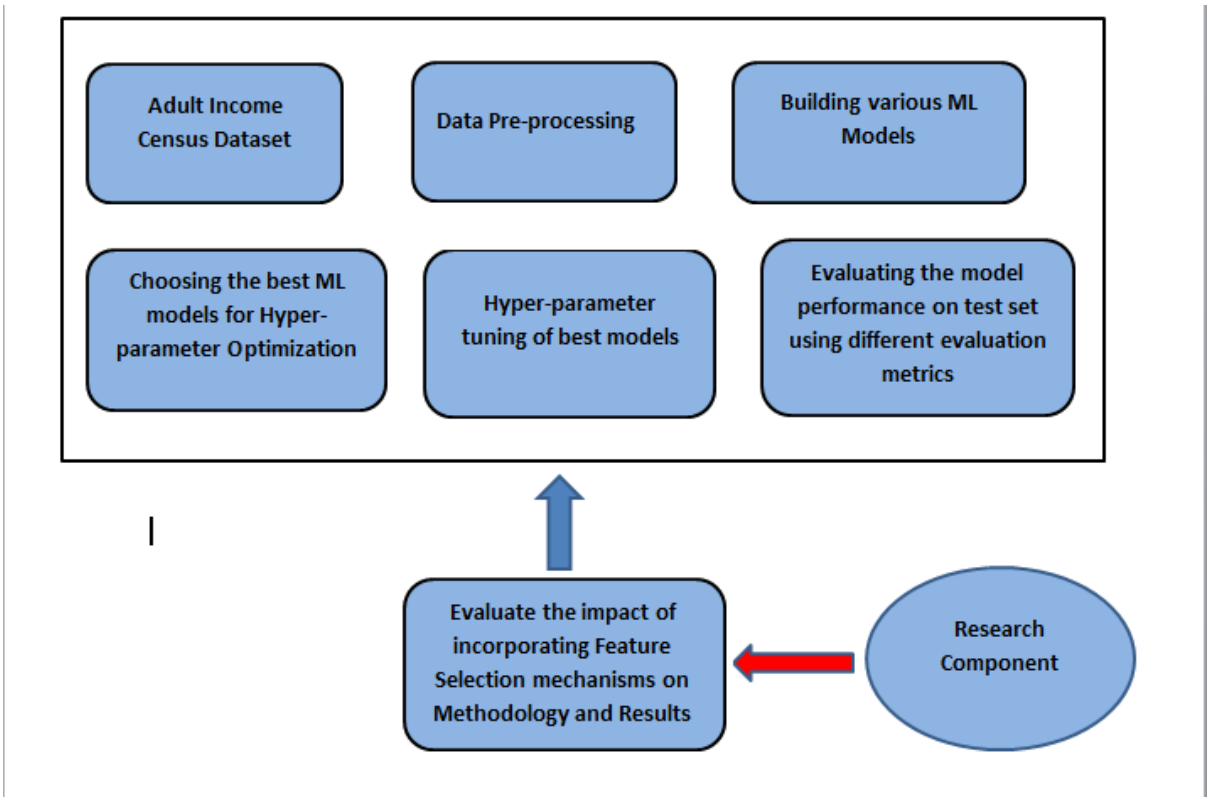


This approach of feature selection using the recursive feature elimination mechanism automatically reduces the number of features involved in learning model based on their effective contribution to the overall accuracy performance of the algorithm. In this technique the external estimator assigns the weight to each of the feature by building the feature importance array and recursively consider the smaller and smaller set of features to select the most important features. The steps for finding the best feature subset

- First the estimator is trained on the initial set of features and the importance of each feature is obtained either through coef attribute or through feature importance attribute.
- Then the least important feature is pruned from the current set of features.
- This procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached (or until all features have been ranked).

3. METHODOLOGY

The high level workflow of the methodology undertaken for this research project is



Adult Income dataset: This dataset is obtained from UCI machine learning repository and is in the raw form.

3.1 Steps involved in Data Preparation

Dataset need to go under series of pre-processing steps as a valid input for machine learning algorithms in Scikit learn [4] which has been extensively used in this research project.

1) ***Outlier Detection***: I performed the Univariate outlier detection for the continuous features which are age, fnlwgt , hours-per-week, capital-gain and capital-loss. For the age, fnlwgt and hours-per-week the outliers just appeared outside the whisker boundaries in a line so I didn't considered them as the outlier and did not removed them. For the capital-gain and capital-loss there are very few outliers detected so for the initial exploration we did not remove them as well.
2) ***Dealing with missing values***: Only categorical features work-class (5.73%), occupation (5.75%) and native-country (1.75%) are having the missing values. Only 5% of the total instances in the dataset consist of missing values so we chose to delete these rows.
3) ***Feature Encoding*** (categorical features): There are total of 9 categorical features in this dataset. 2 of them (education and income) are ordinal and 7 (workclass, marital-status, race, gender, native-country, relationship and occupation) are nominal categorical features. I encoded these categorical features into the unique integer representation using label encoding feature of Scikit learn. I dropped the educational-num column as it is the exact same representation as of education feature after being encoded.
4) ***Feature Scaling:*** It is always better for the machine learning and optimization algorithms that all the features in the dataset are on the same scale. I chose to standardize all the features such that the mean becomes 0 and standard deviation becomes 1 leading to a Gaussian distribution.
5) ***Handling Imbalance:*** There is an imbalance of the target feature (income range) in our dataset. 27% of the instances belong to the minority class (income>50K) and 73% of the instances belong to majority class (income<=50K). I employed widely used SMOTE (Synthetic Minority Oversampling Technique) on the training data to deal with this problem. With SMOTE, there is the balance maintained in the accuracy of the individual class evaluated using confusion matrix. The unbalanced test data is kept aside for the final model evaluation.
6) ***Feature Selection:*** Initially no feature selection technique was applied as there are only a total of 14 features left in the dataset including the target feature after feature encoding. We proceed with all the 13 features to see the impact of keeping all of them on the final classification performance. Later on as this is the research element so we will explore the above discussed feature selection mechanisms to see their impact on the performance by our best tuned model.

3.2 a) Building various Machine learning models

We built a range of machine learning models available in Scikit learn with default parameters with the prepared adult income dataset from the pre-processing steps. Some of the algorithms I tried are Logistic Regression, K-nearest Neighbour, Support Vector Machines, Naïve Bayes, Decision Tree, Random Forest, Gradient Boosting and Ada Boost.

3.2 b) Choosing the best models for Hyper-parameter Optimization

I compared the mean accuracy and mean f1-score of each of the machine learning models and picked up the most promising models that will undergo the hyper-parameter optimization. K-fold validation is the technique in which the training dataset is randomly divided into k folds where the k-1 folds are used for model training and 1 fold is used for model testing. This procedure is repeated for k number of iterations so as to get the k values of performance measures (accuracy or f1-score). We then take the mean or standard deviation of all the k accuracies to get the average performance of the model on training set.

In stratified 10-fold cross validation, the training dataset is divided into 10 folds (9 folds for training and 1 fold for testing) such that the target class distribution is preserved in each of the 10 folds. We don't want disparity in our folds in terms of class distribution otherwise we won't be able to get the appropriate estimate of our performance measure. Following table 3.1 shows the performance of several models on the training set using the default hyper parameters.

Machine Learning Models	CV Mean Accuracy	CV Mean F1 Score
Logistic Regression	74.37 %	75.31 %
K-nearest Neighbour	77.38 %	79.90 %
Support Vector Machines	75.13 %	78.61 %
Naïve Bayes	76.61 %	75.56 %
Decision Tree	84.00 %	83.77 %
Random Forest	87.31 %	86.72 %
Gradient Boosting	84.43 %	86.51 %
Ada Boost	85.16 %	85.29 %

TABLE 3.1 – Performance of different models on the training set with default parameters

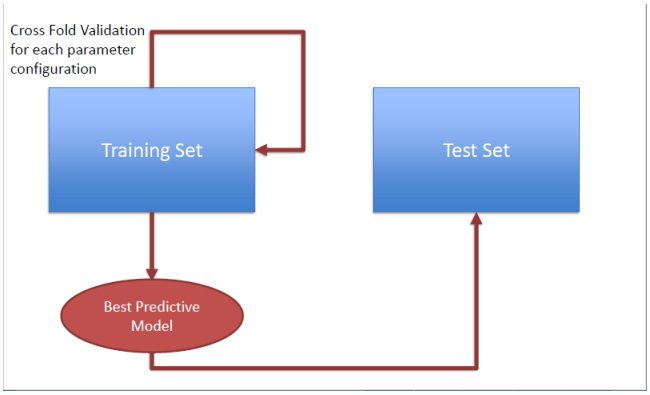
3.3 Hyper-parameter Optimization of Top ML Models

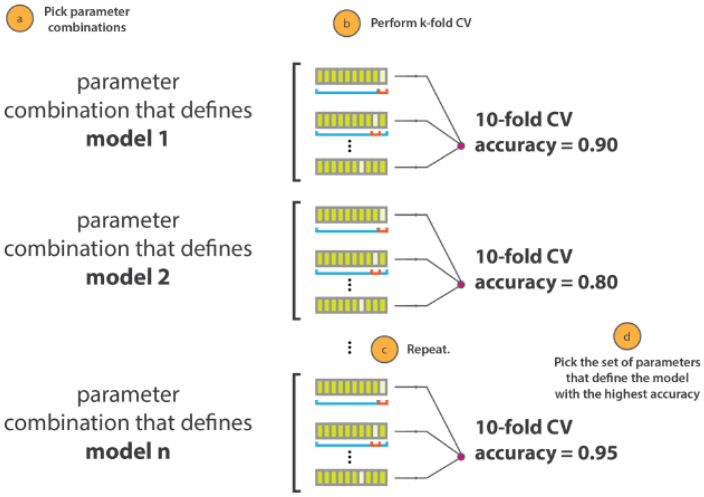
We chose Random Forest and Gradient Boosting for hyper-parameter tuning on the basis of best mean accuracy and mean f1 score on the training set evaluated using the 10-fold stratified cross validation.

Hyper-parameters are the set of arguments within the class of each machine learning algorithm which are not directly learnt by the algorithm on its own. There is a defined

search process for finding out the optimal values of these hyper-parameters which is as follows-:

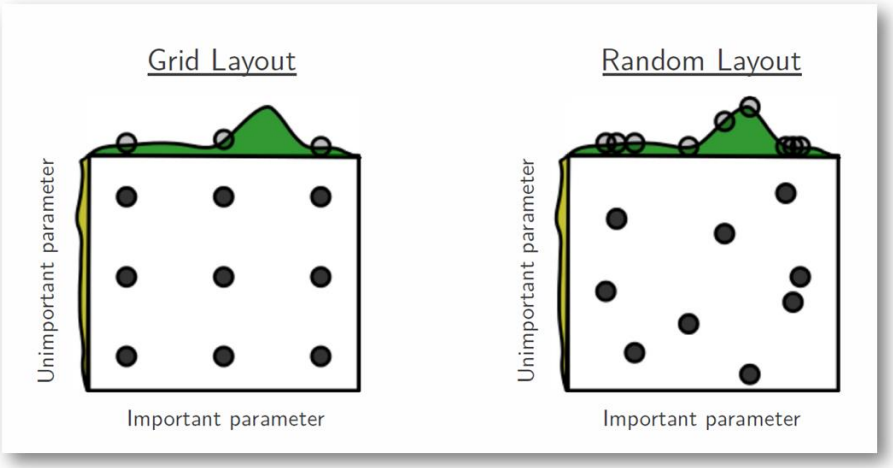
1) A searching algorithm
2) A parameter space (range of values for the parameters)
3) A method for searching the values
4) Cross validation for finding the best hyper-parameters.





We will only perform the hyper-parameter tuning on the training data and test data is kept aside for final evaluation. There would be n different parameter combinations (n different models) depending on the parameter space that the user has given. Each combination of parameter goes through the k-fold cross validation and we obtain the mean accuracy for that combination. The parameter combination which has the maximum (best) k-fold mean accuracy will be chosen as the optimal parameters for that machine learning algorithm.

There are mainly 2 techniques for hyper-parameter tuning- Grid Search CV and Randomized Search CV



Grid Search CV – In this approach, we try every combination of preset values given in the form of parameter grid which is a list of dictionaries. When the hyper-parameters are more, then the number of evaluation increases exponentially with each additional parameter.

Randomized Search CV – Random combinations of hyper-parameters are used to find the optimal parameters. It tries a random range of values given in the form of parameter distribution.

We have applied Randomized Search CV because of the following reasons [5].

- Due to less number of parameter combinations because of random selection, this is faster than Grid Search CV.
- We are tuning the best algorithms for our dataset which are gradient boosting and random forest which has a lot of hyper-parameters so this Random Search CV is more suitable in this case.
- Another reason for using this Random Search CV is that the magnitude of influence of each parameter in both of these algorithms varies.

Table 3.2 shows the performance of top models after hyper-parameter optimization in terms of accuracy and f1 score using 10-fold cross validation.

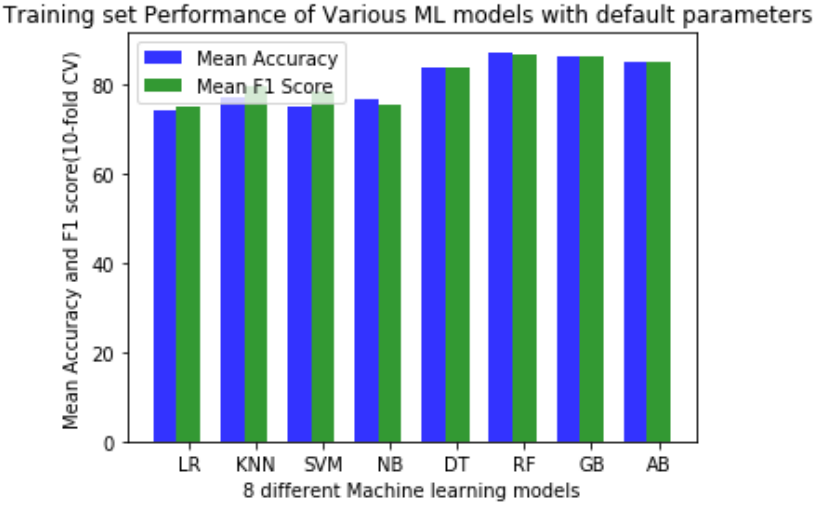
Top ML Models	**Best Mean CV Accuracy**	**Best Mean CV F1 Score**
Random Forest	87.83 %	89.65 %
Gradient Boosting	85.21 %	89.64 %

Table 3.2 Performance of top models on the training set with optimal parameters.

4. EVALUATION

4.1 a) Training set Performance of 8 different Machine learning Algorithms.

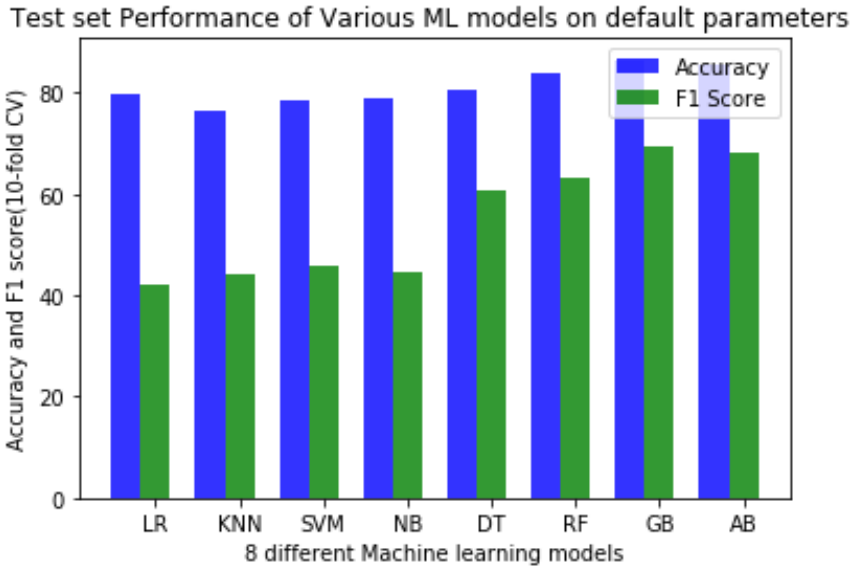
We compared the performance of 8 different machine learning algorithms on the balanced training data. We used 10-fold cross validation on the training data to take out the top models for hyper-parameter tuning. Below graph shows the mean accuracy and mean f1-score (across 10 iterations) for each of the 8 different ML models.



Analysis of the above graph

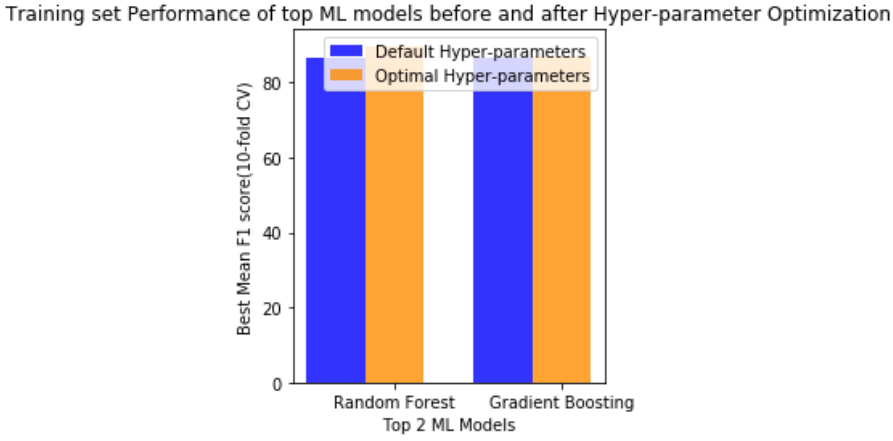
Clearly the most promising models on the basis of the mean f1-score and accuracy are *Random Forest, Gradient Boosting and Ada Boost*. These models will pass through the hyper-parameter tuning but before finalizing these models as the best models we will also evaluate the performance of each of these on the test set. Below graph is the confirmation of the fact that these machine learning models are performing best on the test set as well.

4.1 b) Test set Performance of 8 different Machine learning Algorithms.



4.2 a) Comparison of Performance of top models on the Training set before and after Hyper-parameter Tuning.

I choose to opt out for the Random Forest and Gradient Boosting for hyper-parameter optimization and calculated the optimal parameters using Randomized Search CV technique. We have done whole of the optimization process on the training set and finally found out the best parameter with best mean accuracy and best mean f1 score for each of these top models (Random Forest and Gradient Boosting). Below graph shows the performance of the Random Forest and Gradient Boosting on the training set with default and tuned hyper-parameters.



Best Model	Performance With Default Parameters	Performance With Tuned Parameters
Random Forest	Best Mean Accuracy = 87.31 Best Mean F1 Score = 86.72	Best Mean Accuracy=87.83 Best Mean F1 Score= 89.65
Gradient Boosting	Best Mean Accuracy=84.43 Best Mean F1 Score=86.51	Best Mean Accuracy= 85.21 Best Mean F1 Score= 86.94

Analysis of the above graph

If we take gradient boosting, then there is not much increase in the accuracy and f1 score on the training set after tuning the model. In the case of random forest, there is no significant increase in accuracy after tuning the model, but there is increase in the f1 score by 3%.

4.2 b) Comparison of performance of top models on the Test set before and after Hyper-parameter Tuning using various evaluation metrics.

We used a range of evaluation metrics for evaluating the performance of top models (Random Forest and Gradient Boosting) on the test set. Some of the evaluation metrics are as follows-:

1) Accuracy – It is the measure of the total number of test instances that are predicted correctly by our model. It does not give us insight about the accuracy for the individual classes present in the target feature. When dealing with the imbalance dataset, the model could overall give a high accuracy but the accuracy of the minority class could be very low as compared with the accuracy of the majority class. In our case the test set is imbalanced so we can't rely on the accuracy and need to look at the other metrics for evaluation.

Some Important Concepts: As our problem is the binary classification so we have the confusion matrix broken down by majority class (0) and minority class (1).

- **True Positives (TP)** – Number of instances of majority class that were correctly classified as majority class.
- **False Positives (FP)** – Number of instances of minority class that were incorrectly classified as majority class.
- **False Negatives (FN)** – Number of instances of majority class that were incorrectly classified as minority class.
- **True Negatives (TN)** – Number of instances of minority class that were correctly classified as minority class.

2) Confusion Matrix – It is the representation of the information about the actual and predicted classifications by the classification algorithm. It gives the insight into the accuracy of each of the classes that are present in the target feature meaning that we get the number of test instances for each class that are correctly as well incorrectly predicted by our model.

3) Recall – It is the measure of sensitivity or the true positive rate. It tells how confident we can be that all instances belonging to a specific class have been correctly classified by the model. The formula for the recall for any of class present in the target feature is as

$$Recall\ for\ majority\ class = \frac{True\ Positives}{True\ positive + False\ Negetive}$$

$$Recall\ for\ minority\ class = \frac{True\ Negetives}{False\ Positives + True\ Negetives}$$

4) Precision – It is the measure of the positive predictive value. It tells how confident we can be that any instance predicted as belonging to a certain class actually belongs to that class. The formula for the precision for any class is as follows

$$\text{Precision for majority class} = \frac{True\ Positives}{True\ positive + False\ Positives}$$

$$\text{Precision for minority class} = \frac{True\ Negetives}{False\ Negetives + True\ Negetives}$$

5) F1 Score – It is interpreted as the weighted average of precision and recall for a particular class. For F1 score to be high of any class, both the precision and recall values should be high as well. The best value for f1 score is 1 and worst value is 0.

Formula for F1 score is given as

$$F1\ Score\ for\ any\ class = 2 * \frac{Recall * Precision}{Recall + Precision}$$

6) Classification Report - This report provides an insight into the precision, recall and f1 score broken down by the class. For our problem as the data is imbalanced, so we will not see the normal accuracy, rather our evaluation for this biased classification would be on the basis of confusion matrix (indication of the accuracy of each of the class). Also we would be considering the f1 score across each of the 2 classes which tell about how good the balance is between the precision and recall values. Moreover this classification report also generates the mean and weighted average of precision, recall and f1 score for each of the individual classes in our target feature. So we would be considering the macro average and weighted average of f1 scores for both the classes as the one of the metric for evaluation. Below table shows the results from the classification report before and after the hyper-parameter tuning.

Best Models	Evaluation on test set with Default parameters	Evaluation on test set with tuned parameters
Random Forest	**Accuracy= 83.17** **Results from the Confusion Matrix are -:** **Majority Class Acc=89.03** **Minority Class Acc=65.00** **Majority Class F1= 89.00** **Minority Class F1= 66.00** **Macro F1 score= 77.00** **Weighted F1 score= 83.0**	**Accuracy=83.67** **Results from the Confusion Matrix are-:** **Majority Class Acc=87.35** **Minority Class Acc=72.47** **Majority Class F1= 89.00** **Minority Class F1= 69.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**
Gradient Boosting	**Accuracy= 82.70** **Results from the Confusion Matrix are-:** **Majority Class Acc=84.37** **Minority Class Acc=77.65** **Majority Class F1= 88.00** **Minority Class F1= 69.00** **Macro F1 score= 79.00** **Weighted F1 score= 83.00**	**Accuracy= 83.60** **Results from the Confusion Matrix are-:** **Majority Class Acc=86.15** **Minority Class Acc=75.86** **Majority Class F1= 89.00** **Minority Class F1= 70.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**

Exhaustive analysis of the results on the test set before and after parameter tuning is as below

- Not considering the accuracy as it is not a good indicative of the performance for our census test data because it is imbalanced.
- For the random forest after tuning of the parameters, there is the much improved balance in the accuracy of individual classes when compared with the default parameters. After tuning, the minority class accuracy goes up by 7.47% improving the model performance when compared with the default settings. For the gradient boosting the accuracy for the majority class has increased by 1.78 % and at the same time the accuracy for the minority class decreases by 1.79 % after tuning the model. This means that there is the introduction of some bias in the predictions for the majority class and the imbalance has affected the tuned gradient boosting model little more than its default settings.
- In the case of random forest after tuning, the f1 score for the majority class remains the same but for the minority class increases by 3% leading to an improvement in the performance. For the gradient boosting the tuning has improved the f1 score for both majority and minority class by just 1%.

Final Words about the best model based on the above facts

We choose the **f1 score** as the most appropriate metrics for our census dataset as it is imbalanced. On the basis of f1 score of majority and minority class, the tuned gradient boosting has proven to be the winner for our income range prediction. The weighted average of F1 score for our best model (tuned gradient boosting) comes out to be 84.00 % and macro average of f1 score of both the classes comes out to be 79%.

Now we have integrated various feature selection mechanisms into our pipeline and evaluated the impact of each of them on our test set. We compared the performance before and after feature selection with our tuned random forest and tuned gradient boosting models.

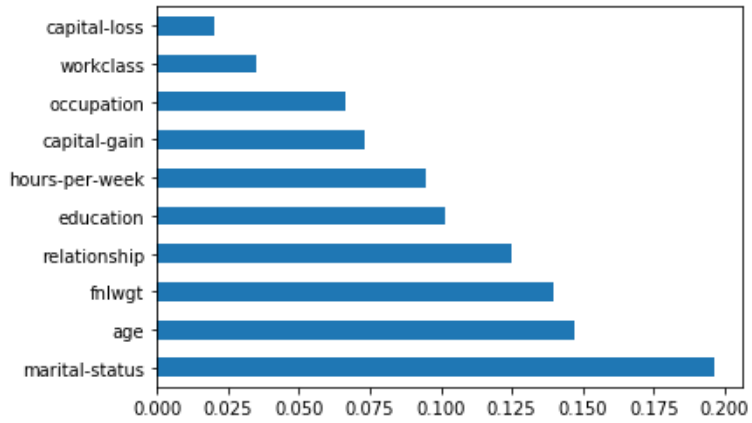
4.3 a) Comparison of test set performance before and after Univariate Feature Selection.

Model Name	Without Univariate Feature Selection	With Univariate Feature Selection
Tuned Random Forest	**Results from the Confusion Matrix are-:** **Majority Class Acc=87.35** **Minority Class Acc=72.47** **Majority Class F1= 89.00** **Minority Class F1= 69.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**	**Results from the Confusion Matrix are-:** **Majority Class Acc=87.15** **Minority Class Acc=72.52** **Majority Class F1= 89.00** **Minority Class F1= 69.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**
Tuned Gradient Boosting	**Results from the Confusion Matrix are-:** **Majority Class Acc=86.15** **Minority Class Acc=75.86** **Majority Class F1= 89.00** **Minority Class F1= 70.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**	**Results from the Confusion Matrix are-:** **Majority Class Acc=85.53** **Minority Class Acc=76.80** **Majority Class F1= 89.00** **Minority Class F1= 70.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**

There was no great significant impact on the test set with Univariate feature selection for both of the tuned models when compared with the performance without feature selection. For the tuned gradient boosting after Univariate feature selection, the majority class accuracy was decreased from 86.15 to 85.53 and the minority class accuracy increased by approx. 1%. All other evaluation metrics remained unchanged.

4.3 b) Comparison of test set performance before and after Tree Based Feature Selection.

The feature with the highest feature importance has the highest impact on the classification. We have used the random forest classifier to extract the top 10 features based on the feature importance and removed the lowest 3 features which were race, native country and gender. Below are the top 10 features



Model Name	Without Tree Based Feature Selection	With Tree Based Feature Selection
Tuned Random Forest	Results from the Confusion Matrix are-: Majority Class Acc=87.35 Minority Class Acc=72.47 Majority Class F1= 89.00 Minority Class F1= 69.00 Macro F1 score= 79.00 Weighted F1 score= 84.00	Results from the Confusion Matrix are-: Majority Class Acc=87.52 Minority Class Acc=71.90 Majority Class F1= 89.00 Minority Class F1= 69.00 Macro F1 score= 79.00 Weighted F1 score= 84.00
Tuned Gradient Boosting	Results from the Confusion Matrix are-: Majority Class Acc=86.15 Minority Class Acc=75.86 Majority Class F1= 89.00 Minority Class F1= 70.00 Macro F1 score= 79.00 Weighted F1 score= 84.00	Results from the Confusion Matrix are-: Majority Class Acc=86.28 Minority Class Acc=75.91 Majority Class F1= 89.00 Minority Class F1= 70.00 Macro F1 score= 79.00 Weighted F1 score= 84.00

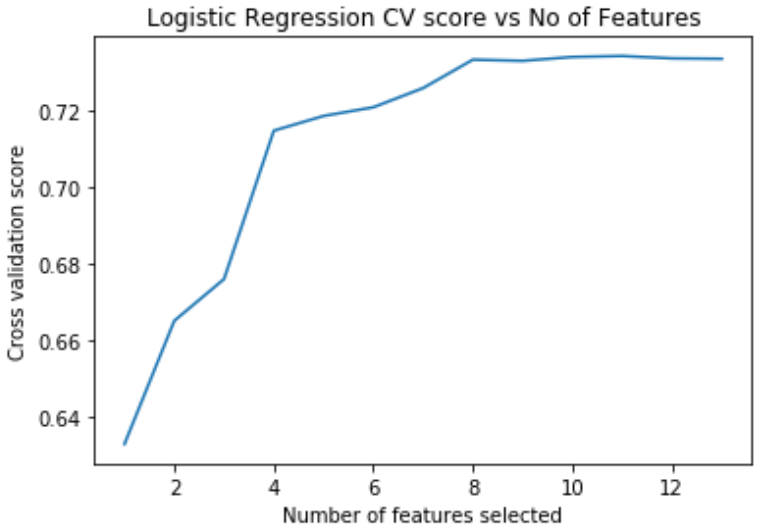
For the tuned random forest model after tree based feature selection there is a little bit decrease in the minority class accuracy by 0.57% obtained from the confusion matrix and other metrics remained unchanged. The performance measure with the tuned gradient boosting remained more or less same with and without tree based feature selection.

4.3 c) Comparison of test set performance before and after feature selection using the Correlation Matrix Heat-map

We just looked at these coefficient values of every feature with the target variable. For extraction of top 10 features we just removed the 3 features that have the coefficient values closer to 0. In our case these were final weight (-0.0065), workclass(0.016) and native-country(0.02) that are least correlated with the target variable income. For both of the tuned models there was no change in the evaluation metrics after doing feature selection using correlation matrix.

4.3 d) Comparison of test set performance before and after feature selection using the Greedy Feature Selection (Recursive feature elimination with cross validation)

Below graph shows that the RFECV algorithm with the logistic regression classifier selected the combination of most important 11 features that gave the highest CV accuracy and we removed the final weight and native country features that were least important to see the impact on the test set.



Model Name	Without RFECV Feature Selection	With RFECV Feature Selection
Tuned Random Forest	**Results from the Confusion Matrix are-:** **Majority Class Acc=87.35** **Minority Class Acc=72.47** **Majority Class F1= 89.00** **Minority Class F1= 69.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**	**Results from the Confusion Matrix are-:** **Majority Class Acc=86.90** **Minority Class Acc=75.11** **Majority Class F1= 89.00** **Minority Class F1= 70.00** **Macro F1 score= 80.00** **Weighted F1 score= 84.00**

Tuned Gradient Boosting	**Results from the Confusion Matrix are-:** **Majority Class Acc=86.15** **Minority Class Acc=75.86** **Majority Class F1= 89.00** **Minority Class F1= 70.00** **Macro F1 score= 79.00** **Weighted F1 score= 84.00**	**Results from the Confusion Matrix are-:** **Majority Class Acc=86.10** **Minority Class Acc=77.69** **Majority Class F1= 89.00** **Minority Class F1= 71.00** **Macro F1 score= 80.00** **Weighted F1 score= 84.00**

With the tuned random forest after extraction of top features with RFECV technique the minority class accuracy from confusion matrix goes up by 2.64 %. For the tuned gradient boosting after RFECV the minority class accuracy goes up by 1.83%.

5. CONCLUSION AND FUTURE WORK

Initially all the data pre-processing steps were applied to census dataset to make it a valid input for various machine learning and optimization algorithms. Out of a range of machine learning algorithms, random forest and gradient boosting were among the top performing models that went under hyper-parameter optimization using Randomized Search CV method. Both of the models showed some improvement in terms of performance on the test set after parameter tuning. As this is the imbalanced dataset we have chosen F1 score and the insights from our confusion matrix as the most appropriate metrics for the model evaluation. Overall the tuned gradient boosting classifier proved to be the winner for income classification with the F1 score of 84%. We then integrated various feature selection mechanisms into our pipeline and compared the performance on the test set with tuned random forest and tuned gradient boosting. All the feature selection techniques didn't show the significant improvement as the numbers of features are very less in our dataset. Recursive feature elimination with cross validation improved the model performance on the minority class by 3% and 2% on random forest and gradient boosting respectively.

Some of the possible areas of future work could be from either data-preparation phase or hyper-parameter tuning of the various models. Different techniques for handling the missing values like imputation can be tried to investigate on the performance metrics. We could also research on handling the high imbalance in the census dataset with various oversampling and under-sampling techniques. Some more feature selection techniques could be explored like a combination of genetic algorithms and iterative local search for selecting the best features. Some embedded methods for feature selection like LASSO and RIDGE regression could also be explored. We could try grid search CV technique and can try out the various combinations of hyper-parameters to improve the performance of our machine learning classifier in predicting the income range of the individual based on certain features.

References

[1] Alina Lazar: "*Income Prediction via Support Vector Machine*", International Conference on Machine Learning and Applications - ICMLA 2004, 16-18 December 2004, Louisville, KY, USA.

[2] S.Deepajothi and Dr. S.Selvarajan: "*A Comparative Study of Classification Techniques On Adult Data Set",* International Journal of Engineering Research Technology (IJERT), ISSN: 2278-0181 Vol. 1 Issue 8, October2012.

[3] Sisay Menji Bekena:"*Using decision tree classifier to predict income levels*", Munich Personal RePEc Archive 30th July, 2017

[4] "*Scikit-learn: Machine Learning in Python"*, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[5] https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318

[6] https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/