Assignment II – Income Evaluation using Machine Learning

-Sriranjani Sridharan (R00182510)

Abstract The goal of this project is to investigate and build a machine model that would at most accurately predict whether a person's income is greater than 50k (>50k) or less than equal to 50k (<=50k). The profile and income information of 32561 people were used as the baseline data to train the model. The data will go through a set of pre processing steps in order to improve the data quality and to provide an insightful prediction. Following this, the processed data were evaluated with various machine learning models and the results were recorded to determine the top 3 baseline models. The top performing models were further tuned by adjusting the model parameters to improve its performance and yield better results. In addition to this, the importance of various attributes/features involved in the prediction were also studied using relevant feature selection techniques. Throughout the study, the Gradient Boosting classifier has shown promising results with respect to model performance.

1 Introduction

Income classification system helps to segregate the population of a nation into low- and high-income categories which will be vastly used by economists to analyze the trends and patterns in the data. It is also widely used in marketing applications to price customer-centric products and in many other financial organizations for decision making. The dataset used in this project was retrieved from https://www.kaggle.com/lodetomasi1995/income-classification and contains information/features about a person such as age, education, work type (Private/Government/Self-Employed), marital-status, native country, capital gain/loss, race, sex etc. The final column named 'income' will contain a target label of whether the person falls in the '>50k' category or '<= 50k' category. Since the target data to be predicted is labelled, this is a classification problem. The dataset will be processed/cleaned to remove noisy data and will then be fed to the machine learning models for training. The trained models can be used to predict the income category of a newly added person to the database based on the

given information. The dataset has been previously used for a lot of work, some of which are [2], [3] and [4].

2 Methodology

In this section, we will discuss the varying approaches that were followed in pre-processing the data and training the machine learning models. The training process followed was a 3-stage approach, where the pre-processed data was trained using 8 different classification models initially and the top three performing models were evaluated and selected based on its generated accuracy. Each classification model involves many hyper parameters to help with the learning process and can be adjusted manually to fine tune its performance. The top 3 models that were selected underwent hyper tuning to determine the optimal parameter configurations of the best performing model. Each subsection below discusses the methods involved in detail.

2.1 Pre-processing

In machine learning, data preparation is a crucial step as it helps us extract the most valuable knowledge from the data and filter/transform the less important ones. In a few cases, it also involves the conversion of data into a machine-readable form such as encoding/transforming text as computer understandable binary numbers. There are a few problems that are often encountered during data preparation or pre-processing [5]:

- Data contains impossible/unlikely values
- No values recorded (Missing values) in certain fields
- Presence of irrelevant data

Unlikely values can arise due to typo errors or mis-communicated data, which either can be re-entered or be considered as missing values/unknown values or can be removed from the data. Detecting outliers is also a possible way to identify unrealistic data.

Incomplete data is a prevalent problem in real world data sources and can be unavoidable. In order to handle the missing data, numerous approaches have been put forth by researchers, such as imputing/replacing it with the most frequent values or removing the data instances associated with it or by mean/median substitution or by ignoring the whole feature if a larger proportion of it is unknown.

Irrelevant data can be handled by feature selection techniques which follows an approach to quantify the contribution of each feature in the data in predicting the target class outcome. A broad range of feature selections were studied in this project and is discussed in detail in the Section 3.

Although there are many other problems encountered in data preparation such as data imbalance with respect to class labels, curse of dimensionality etc., in this project we will focus more on above mentioned characteristics.

The first step in preprocessing, is to understand the data in hand and its semantics. The helps in determining the right approaches towards preparing the data for model training. The income evaluation dataset contains categorical (text) and numerical data and hence would require transformations/encoding before feeding it into the training models. The distribution of data with respect to the class 'income' is shown in Figure 1. The graph shows an unequal distribution, however the impact of it was much less on the performance of the models, hence the class distribution was kept as is throughout this study.

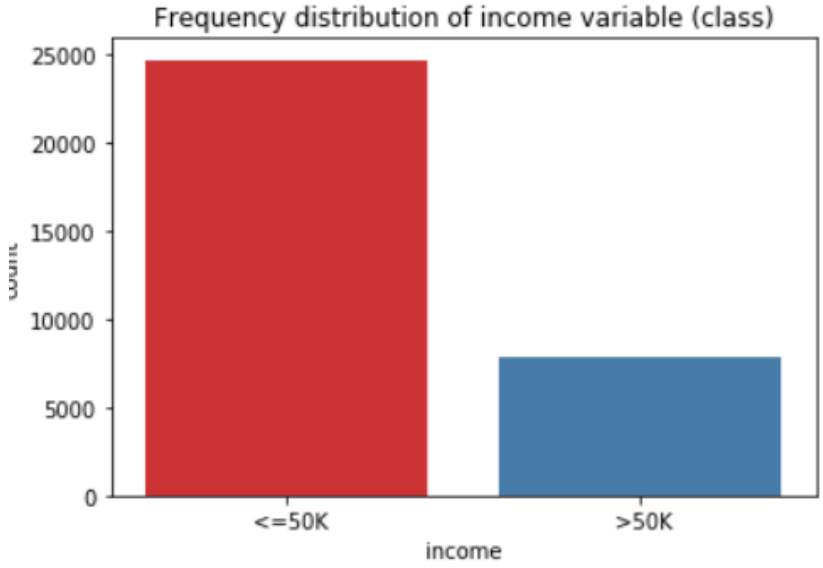

Figure 1

The next step was determining the unknown or incomplete values in the dataset. The fields 'workclass', 'occupation' and 'native_country' contained unknown values represented by '?', which was converted to missing values as understood by the system. Certain machine learning algorithms can handle missing data however, the predictions do not turn out to be reliable. To overcome

this issue, the missing values were replaced by the most frequent value in the feature set as the features being dealt are of type categorical.

While processing the missing values for the field 'native_country', it was observed that 91% of it had 'United -States' while all other countries made up to a small proportion (Refer Figure 2). Due to this, the field was converted to a binary variable, holding 1 if the country is 'United-States' or 0 if it any other country. This will help in converting the categorical into a binary field and thus reducing the unnecessary dimensions that will be created when the variable in encoded from text to numbers.

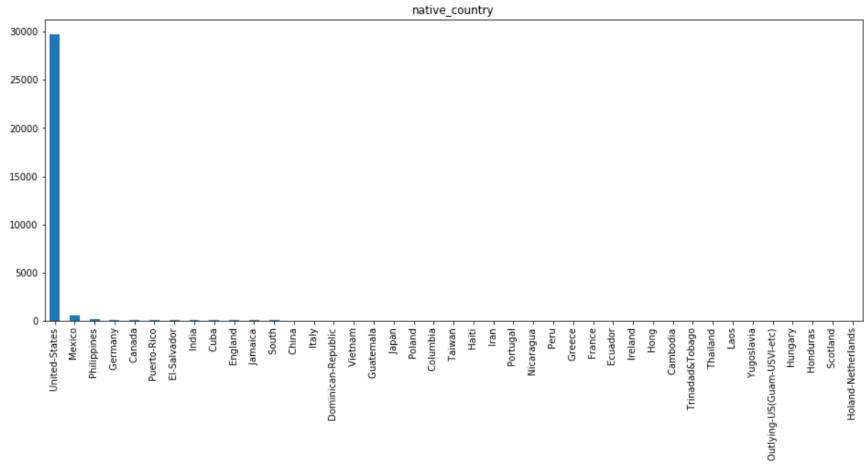

Figure 2

Although the numerical fields are a valid input to the training models, their existing range of values/numbers might be different for different fields and as a result, the features with a larger range tend to dominate the overall predictions. To address this issue, the numerical features in the income dataset was scaled using 'MinMaxScaler', which rescales/normalizes all the fields to have a range between 0 and 1 thus giving equal weights to all the numerical features in the prediction process. On the other hand, the categorical variables will be transformed to numbers using encoding techniques such as Ordinal encoding or One hot encoding. In the current scenario, one hot encoder was used for the transformation which resulted in transforming the 14 features into 65 features. The class labels were separately transformed to numerical by replacing the categories '<=50k' as 0 and '>50k' as 1.

As mentioned before, unlikely/irrelevant values are a common issue in most datasets especially when the data has continuous values. To visualize and detect such outliers, the numerical field values were individually plotted using boxplots (Refer code section - 'Outliers'). Although outliers were detected in all the plotted features, it was not required to remove/ transform all of it. However, in the

'capital_gain' field certain dominant outliers were filtered thus reducing the size of the dataset from 32561 to 32402 instances.

2.2 Models

The prepared data was trained with 8 different classification models such as Decision tree, Linear SVC (SVM with linear kernel), Random Forest, SVM, Naïve Bayes, K Nearest neighbors, Logistic Regression and Gradient Boosting. All the models were trained with default parameter settings and will form the baseline for performance evaluation. Stratified cross fold validation was used to train all the baseline models, where the internal train and test splits for each fold have the same proportion with respect to the target class values as it was in the whole dataset (shown in Figure 1). The evaluation of the results from these baseline models is discussed in detail in the following section.

The top performing models were Linear SVC, Random Forest and Gradient Boosting, which was further tuned based on its hyperparameter to improve its performance. All the hyperparameter optimized models were implemented using GridSearchCV that performs and exhaustive search over a range of parameter combinations. The performance from the best parameters set was recorded for evaluation.

For the Linear SVC model, the most important hyperparameter that was studied over a range of values is the Regularization parameter 'C' and the 'intercept_scaling' parameter. This parameter determines how much of misclassification should be avoided with respect to optimizing the margin width of the separating hyperplane [6]. The intercept scaling is inversely proportional to the effect of regularization. The range of values studied are shown below.

```
param_grid= [ {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
               'intercept_scaling': [2, 4, 5, 10]} ]
```

The random Forest classifier was hyper tuned based on a range of values for max_features, n_estimators, max_depth, min_samples_leaf and min_samples_split. The most important settings are the n_estimators which is number of trees in the forest and max_features which determines the number of features considered for splitting at each node [7]. From the remaining parameters, max_depth determines the maximum levels in each tree, min_samples_split denotes the minimum number of data points placed in each node before it is split, min_samples_leaf is number of data points included in a leaf node and bootstrap determines whether data points are sampled with or without replacement. The range of values [7] studied are shown below.

```
        {'bootstrap': [True],
```

```
'max_features': [2, 3],
'n_estimators': [100, 200, 300, 1000],
'max_depth': [80, 90, 100, 110],
'min_samples_leaf': [3, 4, 5],
'min_samples_split': [8, 10, 12]}
```

Being the best among the top 3 performing models, Gradient Boosting Tree classifier was further tuned with a range of parameters as shown below. While creating new trees sequentially, the residual prediction errors from the existing sequence of trees is corrected which might lead to overfitting the model [8]. To avoid this, the learning rate parameter is used to assign a weighting factor for the sequential corrections. The parameter definitions for n_estimators, max_depth and max_features are similar to the Random forest model.

```
{'learning_rate’: [0.15,0.1,0.05,0.01,0.005,0.001]}
{'n_estimators': [100,250,500,750,1000,1250,1500,175]}
{'max_depth': [1,3,5,6,7,9]}
{'max_features': [3,4,5,6,7]}
```

3 Research

This section describes the in-depth research of Feature Selection techniques for the income evaluation dataset and its impact on the top 3 performing models that were obtained from the baseline training models.

3.1 Feature Selection

Feature Selection helps to make sense of each feature’s contribution in the final prediction and the importance of considering it in the training process. The selection strategies can be broadly categorized into filter, wrapped and embedded methods. Applying feature selection techniques to data helps to improve the performance of the model, train the models faster, decreases the memory requirements and reduces the complexity in training a model. The below subsections give an overview of the selection methods available and the implementations of a few in the project.

3.2.1 Filter Methods

The filter methods are further classified as,

- Univariate Feature Selection
- Multivariate Feature Selection

In the univariate selection method, the features are ranked individually without considering its relationship with other features, whereas multivariate feature selection ranks multiple features together.

Univariate filter methods rank the top N number of features by applying a statistical measure based on their correlation with the 'income' class variable. It uses different types of ranking criteria based on fisher score (F-score), mutual information or variance of the feature [10]. Multivariate selection methods can be helpful in removing redundant features from the data by considering the mutual relationship between features. Some examples of multivariate methods are Inconsistency criterion, Minimum redundancy, maximum relevance (mRmR), Correlation-based feature selection (CFS), Fast correlation-based filter (FCBF) [11].

For the current project, the univariate methods involving F-score and mutual information were implemented with Scikit based selection tools like SelectKBest (select K number of best features) and SelectPercentile (Select features with higher scores in the given percentile). The SelectPercentile was used in combination with the scoring functions F-value and chi2. The former results in the determining the features with highest ANOVA F-values and the later determines the features with highest values of chi-squared test statistic. The SelectKBest was used with chi2 and mutual information, where mutual information measures the dependency between variables.

The dataset with transformed/selected features from the above methods were incorporated in the top 3 performing baseline models and the results were recorded for evaluation.

3.2.2 Wrapper methods

Wrapper methods use predefined learning algorithms and its predictive performance to assess the quality of a subset of features. It is designed similar to a search problem, where (1) an initial subset of features is chosen, (2) its performance is evaluated, and the steps 1 and 2 are repeated iteratively until a stopping criterion is reached or until the highest learning performance is achieved [9]. Following are the examples of wrapper method types.

- Forward Selection

Starting with an empty feature set, features are added sequentially at a time based on the performance of the classifier as long as the performance keeps improving.

- Backward Elimination
  Starting with a feature set containing all the features, it gradually removes the least significant feature based on the performance of each iteration until no improvement is observed. [12]

- Recursive Feature Elimination (RFE)
  RFE selects recursively by considering subsets of features in each iteration by assigning weights to it. For every iteration, the least important feature is dropped and this repeats until all the features have been ranked.

In the current project, the RFE method was implemented with Logistic Regression as the estimator to reduce from 65 to 30 features. The transformed dataset with the reduced features was then incorporated with the best performing baseline models for evaluation.

3.2.3 Embedded Methods

Embedded methods combine filter and wrapper methods by using algorithms that have built in feature selection capabilities. Regularization/Penalization is a widely embedded method which targets to fit a model with minimal fitting errors and by forcing a reduction in magnitude of feature coefficients [9]. Examples of Regularization methods are,

- LASSO Regression
  L1 Regularization considering the absolute value of the magnitude of coefficients as penalty [12]

- RIDGE Regression
  L2 Regularization considering the square of magnitude of coefficients as penalty [12]

Other embedded methods use Tree based classifiers using Random Forest, or Extra Tree Classifiers. The relative importance of each feature is a by-product of these ensemble methods and it can be accessed to perform feature selection [13].

The embedded method using Random Forest classifier was implemented and the transformed feature set generated from the model was trained on the top performing baseline models. The evaluation of the results from all the above discussed methods is presented in the following section.

4 Evaluation

The evaluation and results were studied in three stages based on (1) the baseline models, (2) Hyper tuned models and (3) Feature Selection implemented models.

4.1 Baseline Model Results

The data from the preprocessing step was fed into 8 different training models which resulted in the accuracies shown in Figure 3. All the models were executed using default parameters. Since Stratified cross validation with 10 folds was used for all the baseline models, the mean accuracy from all folds were recorded as the model accuracy.

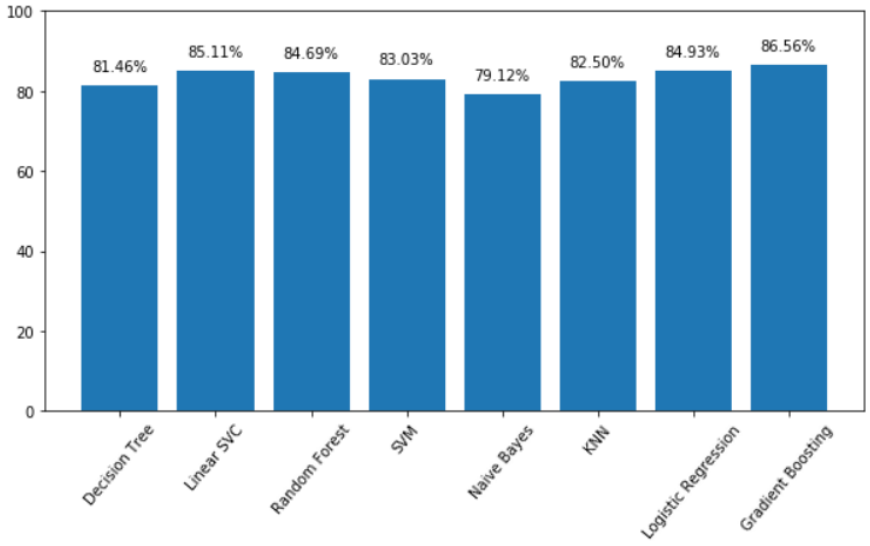

Figure 3. Baseline Model Accuracies

The top performing classifiers follow the order Gradient Boosting > Logistic Regression> LinearSVC > Random Forest, while Naïve Bayes and Decision Tree classifiers comparatively were the weak performers.

The top 3 performing models that were selected for hyper tuning and the feature selection research were Gradient Boosting, Linear SVC and Random Forest.

Top 3 models	Baseline Accuracy
Linear SVC	85.1%
Random Forest	84.7%
Gradient Boosting	86.6%

4.2 Hyperparameter tuned Model Results

A range of model parameters for the selected top 3 baseline models were studied by implementing the GridSreachCV method, to find the best performing parameter settings.

4.2.1 Linear SVC

For the Linear SVC model, the Regularization parameter 'C' and the 'intercept_scaling' parameter were varied as mentioned in Section 2.2. With the same parameter grid, the penalty parameter was varied between 'l1' and 'l2' and the best parameters set for both the executions individually were recorded as given below.

First Execution (penalty l2)

```
Best parameters {'C': 100, 'intercept_scaling': 5}
Accuracy: 0.8512637818248825
```

Second Execution (penalty l1)

```
Best parameters {'C': 100, 'intercept_scaling': 5}
Accuracy: 0.8512637818248825
```

From both the above executions the accuracy was similar although the intercept_scaling for each penalty setting was different. The impact of intercept-scaling has not affected the model performace.

It can be observed that the accuracy from the hyper parameter tuned model is similar to the baseline model which reported 85.1% and has not added value to the performance.

4.2.2 Random Forest

The range of Random Forest parameters that were studied for hyper tuning was mentioned in Section 2.2. The best parameter performance was reported as given below with an accuracy of 85.8%, which was slightly higher than the baseline model accuracy of 84.6%

```
Best parameters
{'bootstrap': True, 'max_depth': 90, 'max_features': 3,
'min_samples_leaf': 3, 'min_samples_split': 8,
'n_estimators':   1000}

Accuracy: 0.8585889759891365
```

4.2.3 Gradient Boosting

Gradient Boosting has been one of the promising models throughout the study and is considered as the top performing model of the three models discussed in this section. The following four executions were studied individually, and the resulting accuracy and best parameter setting are reported. Except the grid parameters the remaining parameters were set as default.

Parameter grid values	**Resulting Accuracy**	**Best parameter from grid**
'learning_rate':[0.15,0.1,0.05,0.01,0.005,0.001]	86.8%	0.15
'n_estimators':[100,250,500,750,1000,1250,1500,175]	87.3%	500
'max_depth':[1,3,5,6,7,9]	87.2%	6
'max_features':[3,4,5,6,7]	86.5%	4

The baseline accuracy was reported as 86.5% and from the above table it can be observed that there has been an improvement with the 1st three parameter tuned models.

A final hyper parameter tuned model with the best grid parameters for learning rate (0.15), n_estimators (500) as indicated above was implemented and the resulting accuracy was 87.3% which is the highest accuracy that was recorded in this project.

4.3 Feature Selection (Research) based Model Results

A couple of methods were implemented and studied from each of the filter, wrapper and embedded methods in feature selection as discussed in Section 3.2. Refer code with below section numbers (4.3.1, 4.3.2 and 4.3.3 for all the evaluation model results).

4.3.1 Filter methods using SelectPercentile and SelectKBest

From the filter method, the impact of univariate chi-2, f_classif statistic was experimented with SelectPercentile tool. The SelectKBased tools was also experimented with chi-2 and mutual-info-classif combinations. The resulting accuracies of top 3 models with the best resulting univariate filter method is shown below in Figure 4.

For the reported feature scores and evaluations results from other methods, refer this report section number (4.3.1) in code.

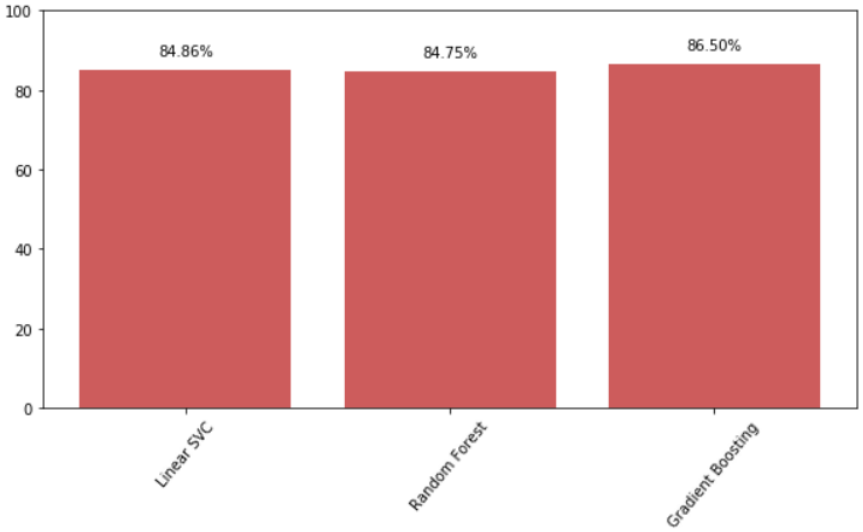

Figure 4 - SelectKBest with chi squared statistic (Accuracy from top 3 models)

The univariate methods have not shown much improvement in the model performance when compared to the baseline models.

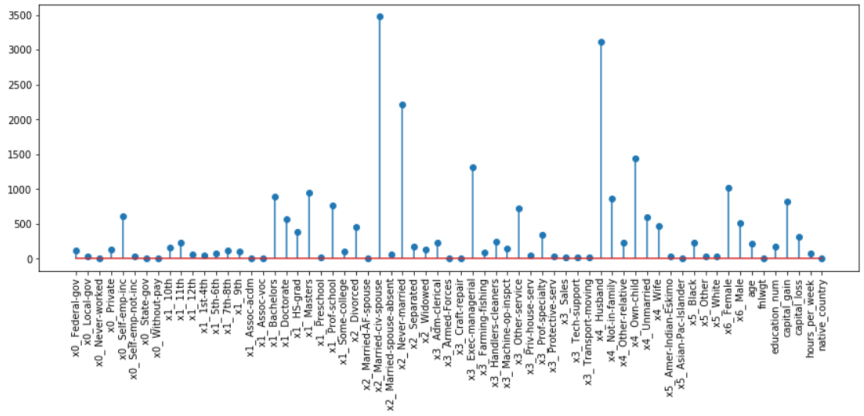

Figure 5 - An overview of the univariate feature selection scores from the encoded array of features

4.3.2 Wrapper methods using Recursive Feature Elimination (RFE)

For this method RFECV was combined with Logistic Regression algorithm to select 30 best features from the encoded feature vector. The resulting accuracies from implementing the reduced data over the top 3 models is shown in Figure 6.

Refer report section in code for details.

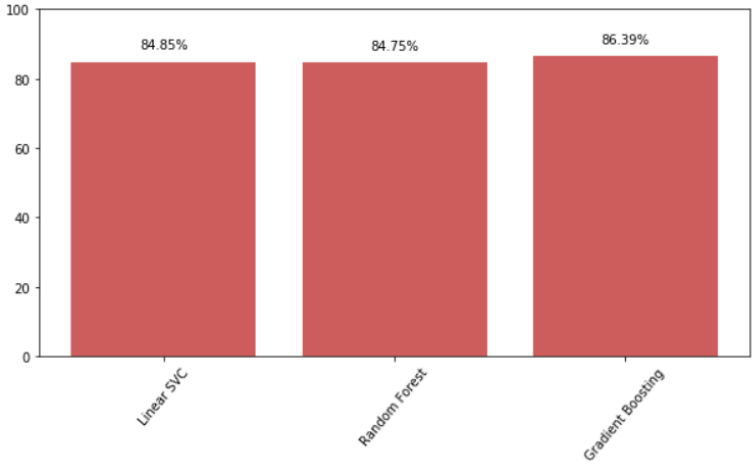

Figure 6 - RFECV with Logistic Regression algorithm (Accuracy across top 3 models)

The results from this feature selection method shows results almost like the baseline accuracy model.

4.3.3 Embedded methods using Tree based classifiers (Random Forest)

Using the Transformer SelectFromModel and the Random Forest Classifer as the estimator, the features are ranked based on the feature_importances_ attribute. This attribute is a byproduct of the Random forest model that was fitted. The feature reduced data was passed thru the top 3 models and the accuracies were recorded as shown below.

Refer report section in code for details.

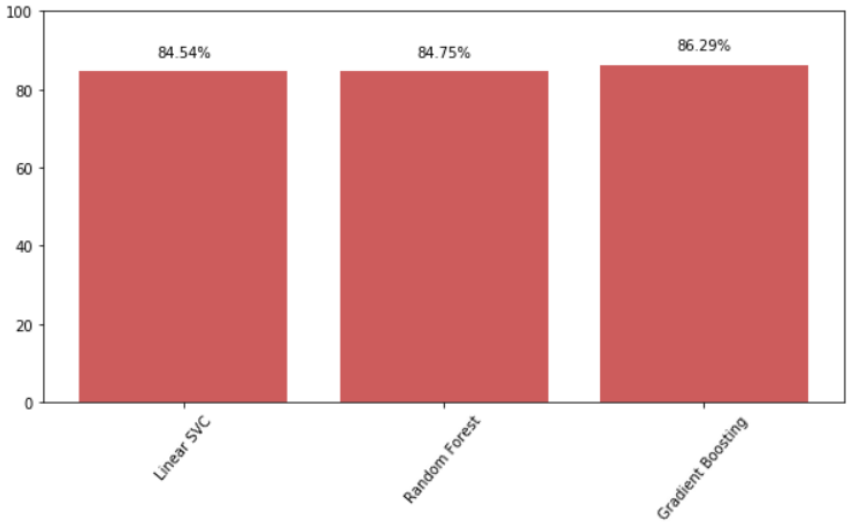

Figure 7 - Tree based method with Random Forest Classifier (Accuracy across top models)

The results from this method is similar to the previous method and hasn't improved much over the Baseline model performance.

5 Conclusion

From all the models that were used in this study, Gradient Boosting has shown the best performance in the income classification process. The reported accuracy for

best performance accounts to 87.4% as given in Section 4.2.3. For the top 3 models, the hyper parameter tuning has shown improved results over the baseline models. However, the results from feature selection methods was almost similar to the baseline models. This might have resulted due to the feature selection from one-hot encoded array of features. The above experiments can be repeated with different feature encoding techniques to see if the feature selection methods show an improvement over the model performances.

References [1]

1. Lecture Notes provided by Prof.Ted Scully
2. Ron Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid", Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996
3. Ke Wang and Shiyu Zhou and Ada Wai-Chee Fu and Jeffrey Xu Yu. Mining Changes of Classification by Correspondence Tracing. SDM. 2003.
4. Dennis P. Groth and Edward L. Robertson. An Entropy-based Approach to Visualizing Database Structure. VDB. 2002.
5. S. B. Kotsiantis · I. D. Zaharakis · P. E. Pintelas : Machine learning: a review of classification and combining techniques (2007)
6. https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel
7. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
8. https://machinelearningmastery.com/tune-learning-rate-for-gradient-boosting-with-xgboost-in-python/
9. Li, Jundong & Cheng, Kewei & Wang, Suhang & Morstatter, Fred & Trevino, Robert & Tang, Jiliang & Liu, Huan. (2016). Feature Selection: A Data Perspective. ACM Computing Surveys. 50. 10.1145/3136625.
10. https://stackabuse.com/applying-filter-methods-in-python-for-feature-selection/
11. A. Jović, K. Brkić and N. Bogunović : A review of feature selection methods with applications
12. https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/
13. https://dkopczyk.quantee.co.uk/feature-selection/