# Comparative Analysis of Classifier Performance on Adult Census Income Dataset

Hassan Mahmood Khan
*Gina Cody School of Engineering and Computer Science*
Montreal, Canada
ha_khan2@live.concordia.ca

***Abstract*—Income disparity is a significant issue that affects the global population at varying levels. Efforts have been put together to curb this threat to improve the socio-economic fabric of our societies. The research community has put in many efforts to identify primary factors contributing to this inequality. In this regard, they have relied on the UCI Adult dataset; derived from a 1994 US Census survey. This paper aims to predict the income brackets of individuals using supervised machine learning based classification algorithms. It presents a thorough comparative analysis of each classifier's performance using multiple evaluation metrics and the impact of hyperparameter optimization on the predictive accuracy.**

***Keywords—machine learning, income, classification, comparative analysis, optimization, k-nearest neighbor, support vector machine, random forest***

## I. INTRODUCTION

In the past decades, the role of data and information has substantially increased in business applications and decision making. With many streams of data, i.e., raw, structured, unstructured etc., now available, as well as the required computational storage and resources to process it – methods such as machine learning and data manipulation have led to fast, automated, and scalable analysis. Their results have enabled effective decision-making and gaining valuable actionable insights.

Machine learning leverages numerous algorithms for distinct applications; these include but are not limited to regression, clustering, dimensionality reduction, feature inference, and many more. However, it is predominantly used for the purpose of classification, where the model predicts data into buckets i.e., classes.

The idea of economic disparity is still looming and a major cause for concern all over the globe. The United States, which is home to millions of people, faces the issue of income inequality, a major challenge as it is detrimental to an individual's socio-economic well-being. The income of an individual is a top indicator of wealth sharing and is generated based on several attributes. To this end, this paper aims to predict the income bracket of individuals based on a variety of features, and presents a holistic comparative analysis between multiple machine learning algorithms through hyperparameter optimization on a binary classification problem.

This paper has been structured as an introduction, literature review, methodology, implementation details, results & analysis, and conclusion.

## II. LITERATURE REVIEW

A multitude of efforts have been made to predict income levels using classification models.

Chakrabarty and Biswas [1] have employed the use of Gradient Boosting Classifier to predict income levels using the UCI dataset and have successfully recorded an impressive 88.16% classification accuracy. Hongzheng [2] has employed the use of neural networks and has additionally applied various data preprocessing on raw data to determine its impact on predictive accuracy.

Topiwalla [3] has implemented a selection of simple and complex classifiers and has conducted a comprehensive analysis of their performance on the datasets. He has also made use of recently developed algorithms such extreme gradient boosting and stacking methods. Zhang, Lemoine, and Mitchell [4] have reflected on the possibility of mitigating unwanted bias in the dataset and how the trained models reflect those. They have made use of their model to predict income brackets whilst diminishing such impact.

Zhu, Whu, and Chen [5] have devised a method for reducing class noise in large datasets. Their approach makes use of tree induction algorithms which are fed subsets of data to construct good rules for the entire dataset. This mechanism enables the detection of mislabeled instances so that they do impact the performance of the model. To this end they utilized the UCI adult dataset and modeled the performance of various classifiers whilst eliminating noise.

Chockalingam, Shah and Shaw [6] have employed multiple classifiers using the adult income dataset and have been able to record the accuracy of their work. The highest classification accuracy reported by their proposed methodology is achieved by Gradient Boosting Classifier.

## III. PROPOSED METHODOLOGY

We begin with the working flowchart of the model.

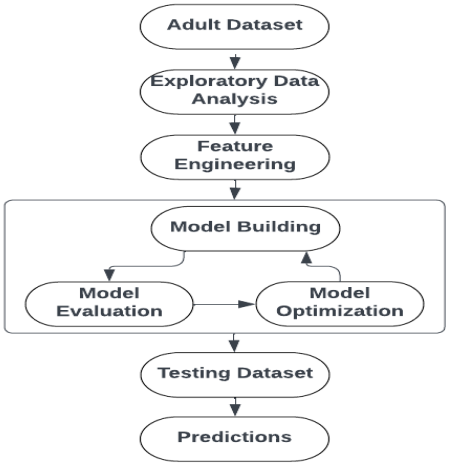


Fig. 1. Model flowchart and its various stages

The following subsections define the main steps and processes undertaken in each phase of the flowchart.

### A. *Dataset*

The dataset for the project is *Adult Census Income*, taken from the University of California Irvine (UCI) Machine Learning Repository [7]. The extraction of data was conducted by Barry Booker from the 1994 Census database. It is multivariate in nature and contains 14 attributes: 8 categorical and 6 numerical in nature. The dependent variable/label 'income' is categorical in nature with two classes i.e., >50K & <=50K, depicting an individual's per annum income. There are 48,842 distinct records in the dataset. Table 1. lists the attributes and their characteristics.

TABLE I.

| ID | Attribute | Type |
|---|---|---|
| F1 | age | continuous |
| F2 | workclass | categorical |
| F3 | fnlwgt | continuous |
| F4 | education | categorical |
| F5 | education-num | continuous |
| F6 | martial-status | categorical |
| F7 | occupation | categorical |
| F8 | relationship | categorical |
| F9 | race | categorical |
| F10 | sex | categorical |
| F11 | capital-gain | continuous |
| F12 | capital-loss | continuous |
| F13 | hours-per-week | continuous |
| F14 | native-country | categorical |

### B. *Exploratory Data Analysis*

Exploratory Data Analysis (EDA) through statistical and graphical visualization techniques allows us to view the various relationships between the input features and the target variable. Additionally, it also helps in discovering anomalies & hidden patterns in the dataset which otherwise may be difficult to detect. This is significant as it lays the groundwork for the model to be constructed on.

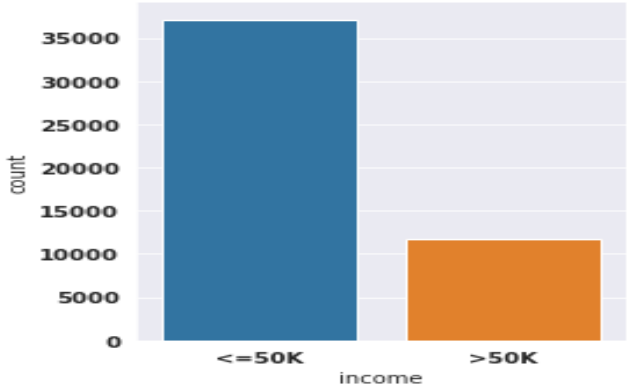


Fig.2. Dataset distribution based on income classes

Viewing the distribution of the target label for all instances using a bar plot, it is immediately clear that our dataset is highly imbalanced with 37155 instances belonging to the '<=50K' class and 11687 instances to the '>50K' class.

We can also recognize the statistical characteristics of our continuous features.

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week |
|---|---|---|---|---|---|---|
| count | 48842.000000 | 4.884200e+04 | 48842.000000 | 48842.000000 | 48842.000000 | 48842.000000 |
| mean | 38.643585 | 1.896641e+05 | 10.078089 | 1079.067626 | 87.502314 | 40.422382 |
| std | 13.710510 | 1.056040e+05 | 2.570973 | 7452.019058 | 403.004552 | 12.391444 |
| min | 17.000000 | 1.228500e+04 | 1.000000 | 0.000000 | 0.000000 | 1.000000 |
| 25% | 28.000000 | 1.175505e+05 | 9.000000 | 0.000000 | 0.000000 | 40.000000 |
| 50% | 37.000000 | 1.781445e+05 | 10.000000 | 0.000000 | 0.000000 | 40.000000 |
| 75% | 48.000000 | 2.376420e+05 | 12.000000 | 0.000000 | 0.000000 | 45.000000 |
| max | 90.000000 | 1.490400e+06 | 16.000000 | 99999.000000 | 4356.000000 | 99.000000 |

Fig. 3. Statistical description of the continuos attributes

All continuous features have been plotted using the Box plot, which enables visualization of the numerical distribution. Similarly, each categorical feature has also been plotted using Bar plots against the target label.

Fig 4. and Fig 5. represent one such example from both feature types.

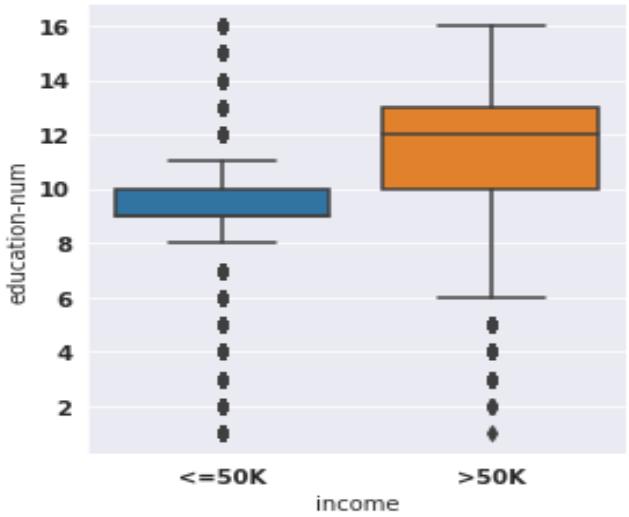


Fig. 4. Education-num vs Income (Box Plot)

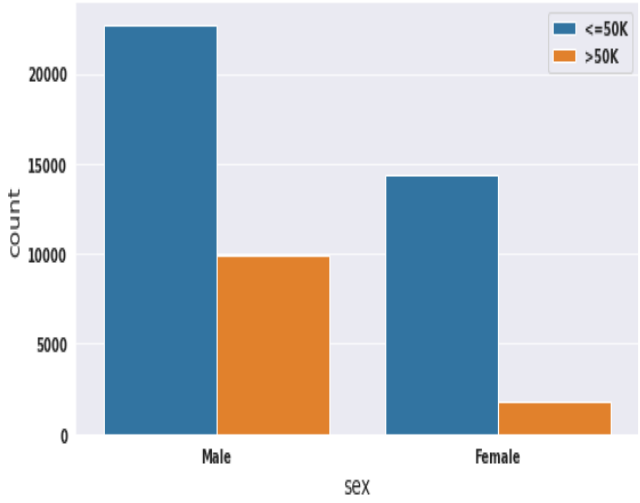


Fig. 5. Sex vs Income (Bar Plot)

### C. Feature Selection & Engineering

The dataset contains only 14 features, and while these are related to the target label, the feature 'education' is the same as 'education-num'. The latter is continuous in nature and imposes ordinality on the values, thus is more representative of the output. In this regard, I have dropped the categorical feature 'education' from our dataset.

The correlation matrix for the continuous features compared with the target variable is shown below.

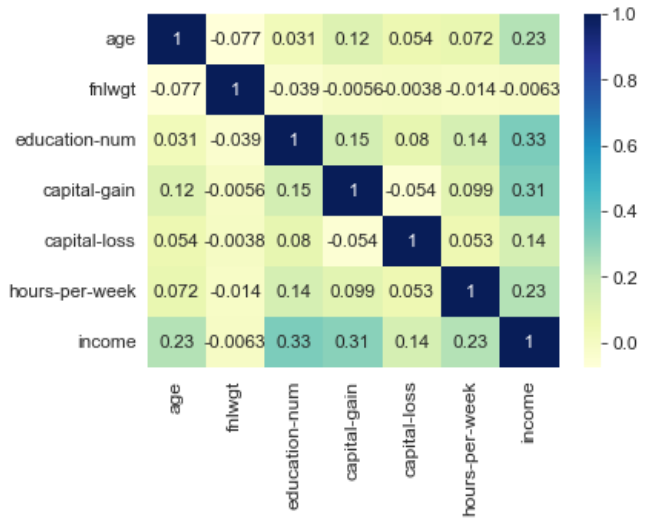


Fig. 6. Correlation Matrix (Numerical Features)

Feature engineering allows us to transform the raw inputs into more predictive features. This aids in interpretability and enhances model's performance. To this end, both features i.e., capital-gain and capital-loss are highly skewed and as such to minimize skewness, I have taken the square root for all instances of these features.

$$capital - gain_{sqr} = \sqrt{capital - gain} \quad (1)$$

$$capital - loss_{sqr} = \sqrt{capital - loss} \quad (2)$$

In addition, the categorical feature 'native-country' is also highly imbalanced with a vast majority of instances belonging to one class i.e., United-States. The table below shows this distribution.

TABLE II.

| Native-Country | Count | Percentage |
|---|---|---|
| United-States | 43832 | 89.75% |
| Rest (41 countries) | 5010 | 10.25% |
| Total | 48842 | 100% |

This presents a challenge since this feature will have a total of 42 classes that need to be transformed into numerical values, imposing a computation overhead, but is also a source of noise due to sparseness. Therefore, I have transformed the feature to have two classes only i.e., United-States & Not United-States. The impact on classification accuracy is discussed in Section IV.

### D. Data Preprocessing

Before the dataset is fed to the model for training and subsequent classification, it is imperative to perform certain preprocessing techniques to clean and transform it into the correct format. These include:

*1) Handling Missing Values:* The dataset contains a total of 3620 instances (7.41%) with missing values, with all values missing belonging to categorical features only i.e., workclass, occupation, native-country etc. Since these missing values carry meaning and give insight into the feature itself, it is important to not discard them to prevent information loss. As a result, I have replaced all such values with a new category, 'Unknown' for all instances.

*2) Column Transformer:* The *ColumnTransformer* class in the scikit-learn python machine learning library allows the user to selectively apply data transformations such as scaling and encoding to different columns in the data frame.

*a) Label Encoding:* Label encoding is a form of categorical encoding that is implemented on the target column. It replaces categorical values into numerical values between 0 and the number of classes minus 1, alphabetically. This helps in interpretability of the dataset and is a requisite for certain algorithms which cannot handle categorical data directly. The table below shows the encoding implemented for this project.

TABLE III.

| Category | Label Encode |
|---|---|
| <=50K | 0 |
| >50K | 1 |

*b) One-Hot Encoding:* Another form of categorical encoding where for each category of a feature, we create a new column and assign a binary value (0 or 1) to denote whether a particular instance belongs to this category. As we are creating additional columns, one for each unique category per feature column, this increase the dimesionality of the dataset significantly. Our dataset contains 7 categorical features, with each feature having a distinct number of categories. The figure below shows the one-hot encoding of one such feature i.e., race.

| race_Amer-Indian-Eskimo | race_Asian-Pac-Islander | race_Black | race_Other | race_White |
|---|---|---|---|---|
| 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |

Fig. 7. Feature 'race' – One Hot Encoded Representation

Each categorical feature has been one-hot encoded apart from the feature 'sex' since it has only two categories i.e., Male & Female, which represents a binary form. This is undertaken to avoid the curse of dimensionality. The dimensionality of the feature set has increased from 14 to 50 – a significant increase.

*c) Feature Scaling:* Certain machine learning algorithms are highly sentive to continuous features that have varying levels of magnitude, range, and units. This can mean certain features are assigned greater weight than the other, introducing bias and can impact the performance of the algorithm. I have employed the use of two feature scaling techniques:

1. *Normalization:* Scale and shift values between a range of 0-1. The formula for normalization is given below:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3)$$

Numerical features 'age', 'education-num', 'hours-per-week' have ordinality, meaning the difference between values represents the distance between each integer. To ensure this property is not lost when scaling – normalization is used.

2. *Standardization:* Values are centered around the mean with unit standard deviation. Standardization does not curtail the range between 0-1, however is effective when there exist numerous outliers. Features such as 'capital-gain', capital-loss', & 'fnlwgt' have been scaled using standardization. The formula is shown below:

$$X' = \frac{X - \mu}{\sigma} \quad (4)$$

$\mu$: mean of feature values
$\sigma$: standard deviation of feature values

*E. Learning Algorithms*

The algorithms employed to build the predictive model are K-Nearest Neighbor, Support Vector Machine, and Random Forest. The following sub-sections discuss them in depth.

*1) K-Nearest Neighbor*: The k-nearest neighbor (KNN) is a supervised, parametric machine learning algorithm. It bases predictions of new data points on the assumption that similar points are found in proximity. Inherently, it is used for classification where a class label is assigned based on a majority vote but can also be used for regression problems. The pseudocode for the algorithm is as defined:

- Compute distance between the new data point and the rest of data points using a distance metric.
- Choose the appropriate distance metric i.e., Euclidean (p=2).
- Euclidean metric formula:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \quad (5)$$

- Sort the distances in a non-increasing order.
- Determine the value for 'k'; how many neighbors to be checked for deciding classification.
- Choose 'k' nearest points based on least distance.
- Assign the majority class of those k points to the new instance.

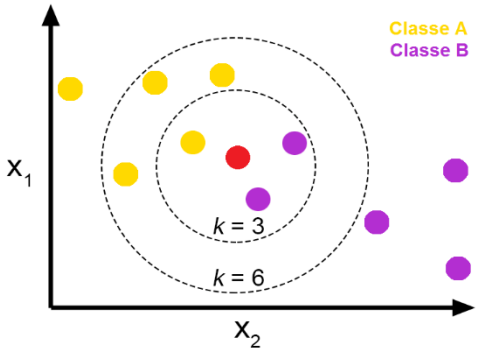


Fig. 8. KNN - Classification. Source (towardsdatascience)

*2) Support Vector Machine:* A supervised, non-parametric model that is employed for classification but can be used for regression. Data points are plotted in an n-dimensional space (where n is the number of features), and a hyperplane is drawn that maximizes the separation between the two classes. There are two types of SVM: linear and non-linear SVM. The pseudocode for the algorithm is as defined:

- Identify the decision surface.
- Transform non-linear decision surface into a linearly separable surface in a higher dimension space using a kernel function e.g., rbf.
- Radial Basis Function (RBF) kernel:

$$K(x - x') = e^{-\gamma||x - x'||^2} \quad (6)$$

- Draw multiple hyperplanes (lines) segregating the two classes.
- Locate data points (support vectors) closest to the hyperplane from both classes.
- Compute distance between support vectors and the hyperplane, known as margin.
- Choose hyperplane that maximizes margin.

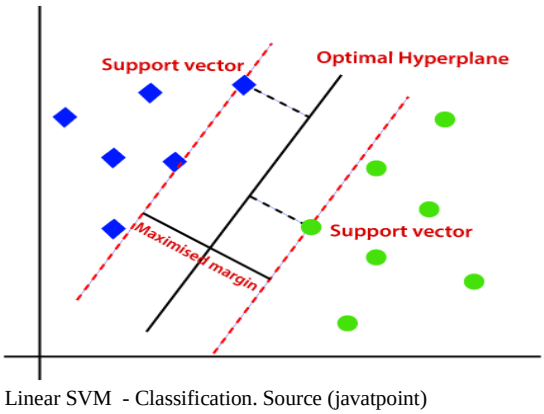


Fig. 9. Linear SVM - Classification. Source (javatpoint)

*3) Random Forest:* Another supervised, non-parametric machine learning algorithm that is widely used in classification and regression problems. It is a bagging ensemble method that combines multiple decision trees models on different training subsets from the training dataset and takes the majority vote for the final classification output. The pseudocode for random forests is as defined:

- 'N' random records are taken from the sample dataset with replacement. (Bootstrap samples)
- Individual decision tree(s) are constructed for each sample.
- The output is generated by each model.
- The final output is determined based on the majority vote for classification, and aggregation for regression.

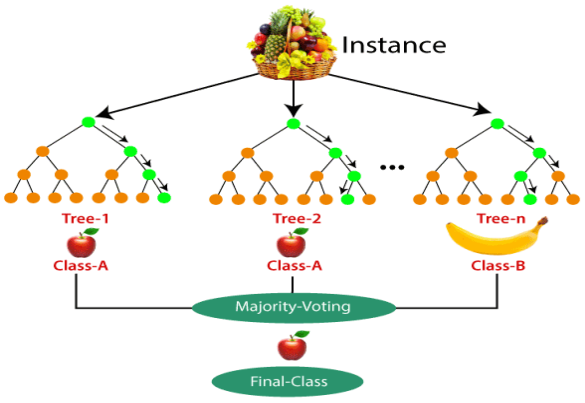


Fig. 10. Random Forest - Classification. Source (analyticsvidhya)

## F. Model Training

All three models are trained and tuned using the following methods:

*1) Stratified K-Fold Cross Validation:* Our dataset faces a large imbalance of the target label as discussed in Section II. The stratified k-fold cross validation is a variation of the k-fold cross validation technique that ensures each fold has approximately the same percentage of target class samples, thus addressing the dataset imbalance to an extent. In addition, it addresses the key issue of overfitting and promotes model generalization.

*2) Grid Search*: The performance of a model significantly depends on the values of the model hyperparamters. I have employed the use of *GridSearchCV* to search all possible combinations of hyperparamter values, to determine optimal values for each of the three models. Important to note, due to time and computational resource restraints, only a selective subset were explored & optimized within a limited range of values.

Table IV represents all hyperparameters investigated and their associated ranges for each classifier.

TABLE IV.

| Model | Hyperparameters | Range |
|---|---|---|
| KNN | 'n_neighbors' | np.arange(1, 50, 2) |
| | 'weights' | ['uniform', 'distance'] |
| | 'p' | [1,2] |
| SVM | 'C' | np.arange(0.1, 1, 0.1) |
| | 'kernel' | ['rbf', 'poly'] |
| | 'gamma' | [0.01, 0.05, 0.1, 0.5, 1.0] |
| Random Forest | 'n_estimators' | np.arange(start=100, stop=150, step=5) |
| | 'criterion' | ['gini', 'entropy'] |
| | 'max_features' | ['sqrt', 'log2'] |
| | 'max_depth' | np.arange(start=2, stop=42, step=2) |
| | 'min_samples_leaf' | np.arange(start = 1, stop = 11, step = 1) |

The figures below depict the summary of *GridSearchCV*.

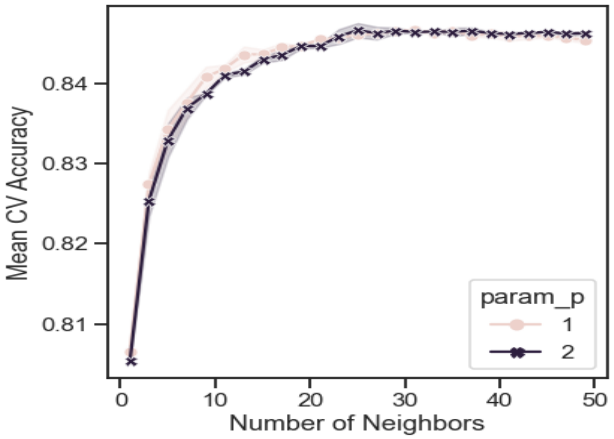


Fig. 11. Grid Search Summary on Mean Cross Validation Score - KNN

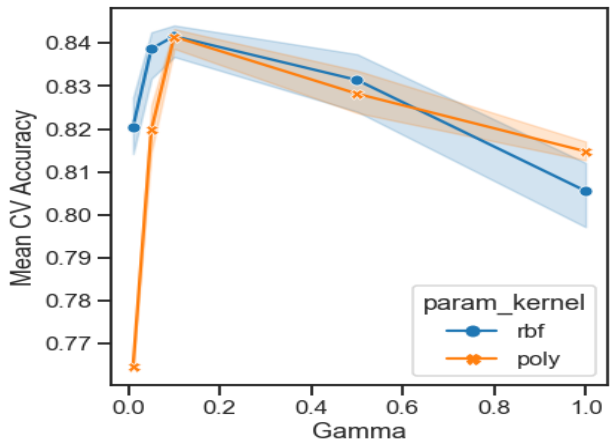


Fig. 12. Grid Search Summary on Mean Cross Validation Score - SVM

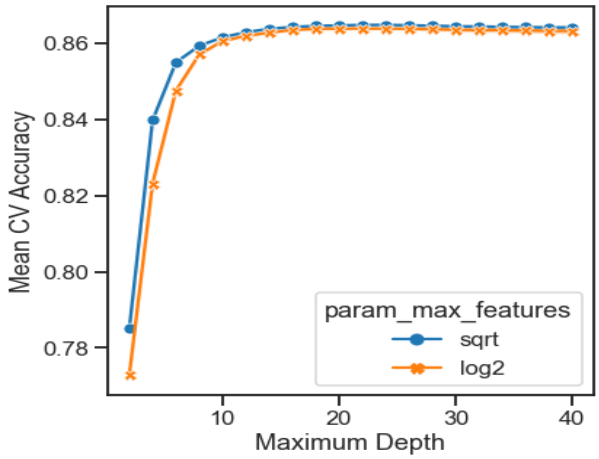


Fig. 13. Grid Search Summary on Mean Score Cross Validation -RF

Following are the optimal hyperparameter values that return best cross validation accuracy for each classifier.

TABLE V.

| Model | Optimal Values |
|---|---|
| KNN | n_neighbors = 25, weights = 'uniform', p = 2 |
| SVM | C = 0.9, kernel = 'rbf', gamma = 0.01 |
| RF | n_estimators = 130, criterion = 'gini', max_features = 'sqrt', max_depth = 24, min_samples_leaf = 3 |

Grid search along with k-fold cross validation imposes a significant computational overhead on the machine. The computational complexity of grid search is O(n), where all combinations of parameters are checked. The computational complexity for k-fold cross validation is O(k*n), where k is the number of folds and n represents the sample size. This is repeated for all three models. Consequently, I have explored a restricted range of values for each hyperparameter based on literature and domain knowledge.

## IV. Implementation Details

This project has been implemented on a 64-bit operating system, x64 based processor (Intel(R) Core (TM) i3-1005G1 CPU @ 1.20GHz  1.19 GHz), with 12.0 GB installed RAM. The IDE employed is Jupyter Notebook and programming language used is Python. All stages of the workflow have been implemented with the use of Scikit-Learn Machine Learning Toolbox, along with additional libraries such as NumPy and Pandas for data manipulation & processing. In addition, Matplotlib & Seaborn have been utilized for data visualization.

## V. Results & Analysis

After obtaining results for hyperparameter optimization, each model is evaluated. Out of the total 48,842 instances, all models are trained on 39, 073 data instances (80%) using a 10-fold stratified cross validation while 9769 data instances (20%) have been reserved for testing.

A thorough performance evaluation procedure is adopted using multiple evaluation metrics.

### A. *Accuracy*

Widely employed for evaluating classification models. Informally, it is defined as the fraction of correct predictions [8]. Formally, it has the following definition:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (7)$$

For binary classification, it can also be defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

Table VI. shows the training & testing accuracy as well as the fit & score time for each model.

TABLE VI.

| Model | Testing Accuracy | Training Accuracy | Fit Time | Score Time |
|---|---|---|---|---|
| KNN | 84.65% | 85.56% | 0.10 s | 25.92 s |
| SVM | 84.86% | 85.25% | 1378.17 s | 24.85 s |
| RF | 86.70% | 90.45% | 8.23 s | 0.90 s |

### B. *Confusion Matrix*

Another technique that summarizes the performance of a classification model. It can overcome any limitations associated with classification accuracy alone and gives insight into the errors being made by the classifier.

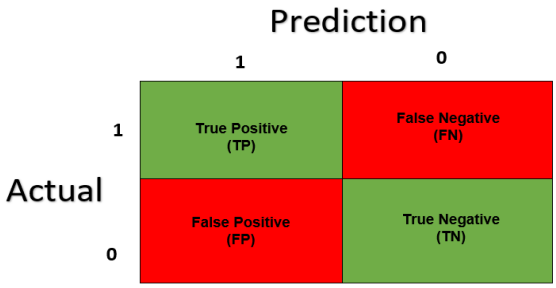


Fig. 14. Structure of Confusion Matrix. Source (TowardsDataScience)

Figures 15, 16 & 17 represent the normalized confusion matrix for each model.

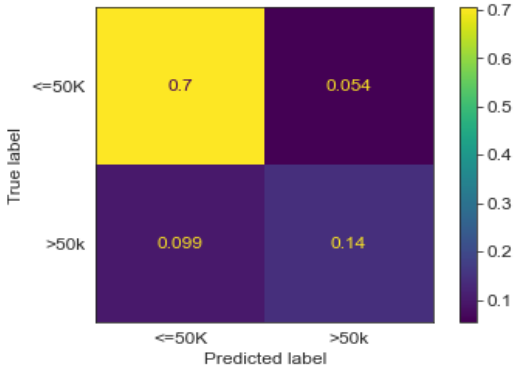


Fig. 15. Confusion Matrix - KNN

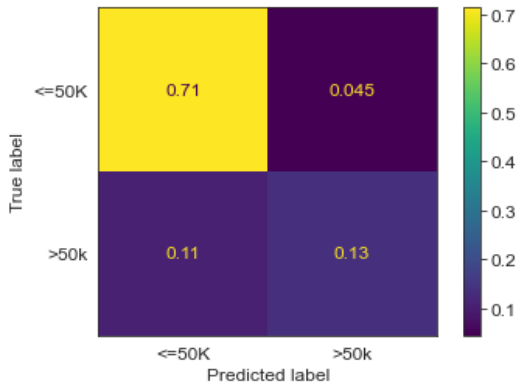


Fig. 16. Confusion Matrix - SVM

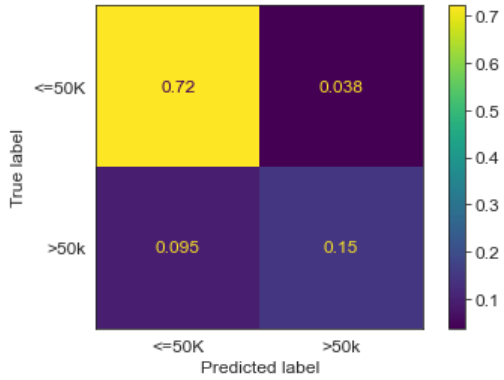


Fig. 17. Confusion Matrix - RF

### *C. Receiver Characteristics Curve (ROC)*

It is a graphical representation of a classifier's performance at all classification thresholds. It plots two parameters: True Positive rate on the y-axis and False Positive rate on the x-axis. In addition, there is another sub-metric that is of importance i.e., Area Under the ROC Curve (AUC). It measures the two-dimensional area under the ROC curve and provides an aggregate gauge of the classifier at all thresholds.

TABLE VII.

| **Model** | **AUC** |
|---|---|
| KNN | 0.888 |
| SVM | 0.890 |
| RF | 0.917 |

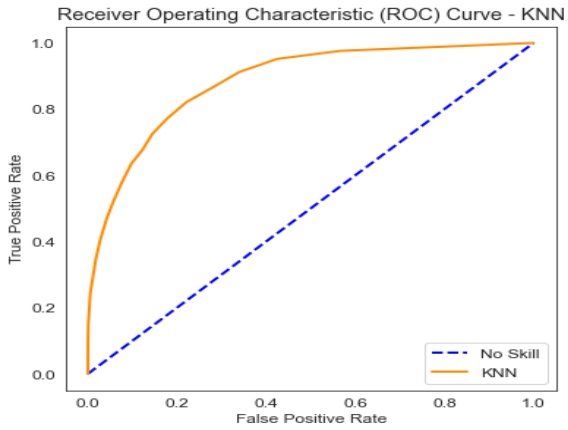


Fig. 18. ROC Curve - KNN

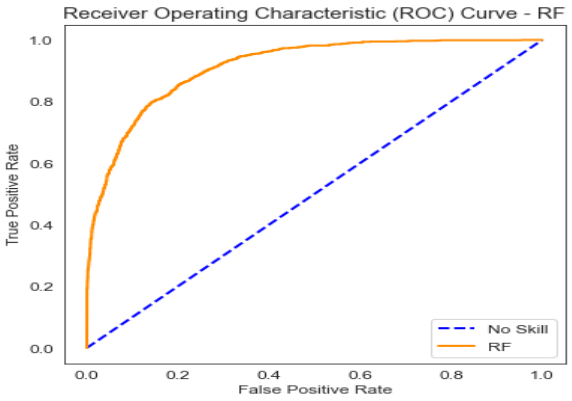


Fig. 19. ROC Curve - SVM

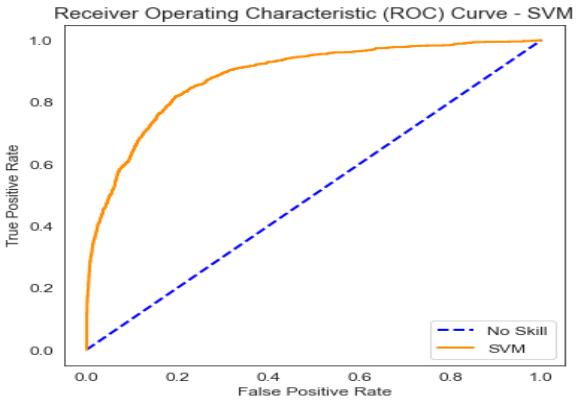


Fig. 20. ROC Curve - RF

Based on metrics computed, it can be observed all models output testing accuracy in a close-knit range of values i.e., 84-86%. Furthermore, the impact of overfitting is greatly reduced due to k-fold cross validation, with training and testing accuracies in neighboring range to each other. A similar trend is observed for AUC scores, with the values varying from 0.88 – 0.91. In addition, we can observe that the ROC curves reside in the top left quadrant of the graph, indicating a high TPR as compared to FPR. We can term these models as skillful, assigning high probabilities to random positive instances than negative ones.

As far as confusion matrix is concerned – the ability of each model to correctly predict instances belonging to class 0 i.e., <=50K, is high. However, the FN rate is considerable and affects the performance of the classifiers. In simple language, this means that individuals earning greater than 50K per annum are incorrectly classified as earning less than 50K per annum. This can be attributed to the previously explored class imbalance and the model's inability to generalize remarkably.

Due to its high dimensionality and a large dataset, KNN did not scale well in terms of computational resources exhausted and running time consumed. The non-linear nature of the decision surface prompted the use of radial basis kernel which computes a NxN matrix of distances calculated. This means the model must convert a 50-dimensional non-linear surface into a linear one, and as expected consumes a lot of time. RF outperforms the other two due to its robustness to noise and simple feature importance calculation.

Random forests classifier is the best performer out of the three classifiers and outputs the highest classification accuracy of 86.70% and an AUC score of 0.917.

A comparison of predictive accuracy obtained with those in literature is represented in the table below.

TABLE VIII.

| Author | Model | Accuracy |
|---|---|---|
| Hassan Khan | K-Nearest Neighbor | 84.65% |
| Hassan Khan | Support Vector Machine | 84.86% |
| Hassan Khan | Random Forest | 86.70% |
| Chakrabarty & Biswas [1] | Gradient Boosting Classifier | 88.16% |
| Hongzheng He [2] | Neural Network (2-layer) | 86.18 % |
| Topiwalla [3] | K-Nearest Neighbor | 79.18 % |
| Topiwalla [3] | Support Vector Machine | 85.52 % |
| Topiwalla [3] | Random Forest | 86.89 % |
| Topiwalla [3] | XGBOOST | 87.53 % |
| Chockalingam, Shah & Shaw [6] | Naïve Bayes | 80.90% |
| Chockalingam, Shah & Shaw [6] | Extra Tree Classifier | 82.12% |
| Chockalingam, Shah & Shaw [6] | Gradient Boosting Classifier | 86.29% |

## VI. CONCLUSION

This paper proposed the application of three supervised, non-parametric machine learning algorithms i.e., K-Nearest Neighbor, Support Vector Machine & Random Forest – aided by data preprocessing and hyperparameter tuning to predict the income levels of individuals in the United States using the Adult Census Dataset [7]. A final validation accuracy of 86.70% is achieved which is at par and above what has been reported in literature.

The metrics reported can be further improved upon inspection and with reliance on better feature engineering methods. Furthermore, the high imbalance between the target classes induces variance into the models, particularly seen in the Random Forest Classifier. This may be reduced through optimal feature selection or by reducing model size/dimensionality of the dataset.

Another concern is the significant consumption of computational resources and time when running the model, in particular the Support Vector Machine. This can be attributed to the high dimensionality of the dataset. Lee and Mangasarian [9] have employed Reduced Support Vector Machines (RSVM) on the adult census income dataset to show how a non-linear kernel-based surface can be obtained using only 1% of a large dataset for its evaluation.

The future scope of this work involves the use of other machine learning algorithms in addition to the existing ones – to improve overall validation accuracy. Moreover, conduct research to apply more effective feature selection techniques that return the optimal set of features. Finally, explore less exhaustive methods for parameter optimization to reduce computational & time overhead.

## REFERENCES


[1] N. Chakrabarty and S. Biswas, "A Statistical Approach to Adult Census Income Level Prediction," 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, pp. 207-212, doi: 10.1109/ICACCCN.2018.8748528.

[2] *H. He, "CLASSIFYING INCOME POTENTIAL FROM THE ADULT DATASET: COMPARING DIFFERENT UNDERSTANDING AND PREPROCESSING DATA.", Research School of Computer Science, Australian National University*

[3] Mohammed Topiwalla: *"Machine Learning on UCI Adult data Set Using Various Classifier Algorithms And Scaling Up The Accuracy Using Extreme Gradient Boosting"*, University of SP Jain School of Global Management.

[4] *Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating Unwanted Biases with Adversarial Learning. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society (AIES '18). Association for Computing Machinery, New York, NY, USA, 335–340. https://doi.org/10.1145/3278721.3278779*

[5] *Xingquan Zhu, Xindong Wu, and Qijun Chen. 2003. Eliminating class noise in large datasets. In Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03). AAAI Press, 920–927.*

[6] Chockalingam, Vidya, Sejal Shah and R. Paul Shaw. "Income Classification using Adult Census Data ( CSE 258 Assignment 2 )." (2017).

[7] https://archive.ics.uci.edu/ml/datasets/Adult

[8] https://developers.google.com/machine-learning/crash-course/classification/accuracy

[9] Lee, Yuh-Jye & Mangasarian, Olvi. (2001). RSVM: Reduced support vector machines. 1. 10.1137/1.9781611972719.13