Finding Donors for Charity

Sarat Kumar Kaniti, Sai Subhash Chandra Reddy Gangireddygari, Kaushal Karinaga Shetter Raju, Abraham Mathew

SAN JOSE STATE UNIVERSITY

May 2022

***Abstract*— The census is a unique, wide-ranging activity that occurs once every ten years across the country. The goal is to collect data on the general public in order to offer a complete and accurate image of the country's population, including housing conditions, demographic, social, and economic features. Age, gender, place of origin, marital status, housing conditions, marriage, education, and employment are among the data collected. The purpose of this machine learning research is to estimate whether or not a person earns more than $50,000 per year based on their demographics. Several classification strategies are investigated, with the Adaboost classification model providing the best prediction result.**

I. INTRODUCTION

The income of the citizens has a significant impact on a country's economic well-being. Many business and public sector choices are dependent on Census data. The democratic form of governance relies heavily on census data, which has a significant impact on the economy.

The government distributes federal cash to different states and localities using census-related data. Not just for the reasons stated above, but also for post-census population estimates and predictions, economic and social scientific research, and a variety of other uses. As a result, the significance of this data and its accurate forecasts is obvious to us. Many crucial judgments have always been based on data. When an assumption is supported by facts and figures, the odds of being wrong and making poor judgments are reduced.

We're using data from a 1994 income survey conducted by the US Census Bureau. With this dataset, we want to solve the problem of identifying persons who make more than $50,000. This information can be useful to non-profit or philanthropic organizations. Understanding an individual's income can assist a non-profit in determining how large of a donation to request, as well as whether or not to contact out in the first place.

A. Dataset

The dataset is collected by the United States Census Bureau's income survey. It contains about 32561 rows and 15 features. The features are as follows:

Feature Name	Type	Description
Age	Numerical	The age of an individual, this ranges from 17 to 90.
Workclass	Categorical	The class of work to which an individual belongs.
Fnlwgt	Numerical	The weight assigned to the combination of features (an estimate of how many people belong to this set of combination)
Education	Categorical	Highest level of education
Education_num	Numerical	Number of years for which education was taken
Marital_Status	Categorical	Represents the category assigned on the basis of marriage status of a person
Occupation	Categorical	Profession of a person
Relationship	Categorical	Relation of the person in his family
Race	Categorical	Origin background of a person
Sex	Categorical	Gender of a person
Capital_gain	Numerical	Capital gained by a person
Capital_loss	Numerical	Loss of capital for a person
Hours_per_week	Numerical	Number of hours for which an individual works per week
Native_Country	Categorical	Country to which a person belongs
Income	Numerical	Income

B. Exploratory Data Analysis

1) Data Preprocessing

We import necessary libraries for the current project. Importing refers to allowing a Python file or a Python module to access the script from another Python file or module. We

can only use functions and properties our program can access. All the standard libraries like numpy, pandas, matplotlib, and seaborn are imported in this step. We use numpy for linear algebra operations, pandas for using data frames, matplotlib, and seaborn for plotting graphs. The dataset is imported using the pandas command read_csv().

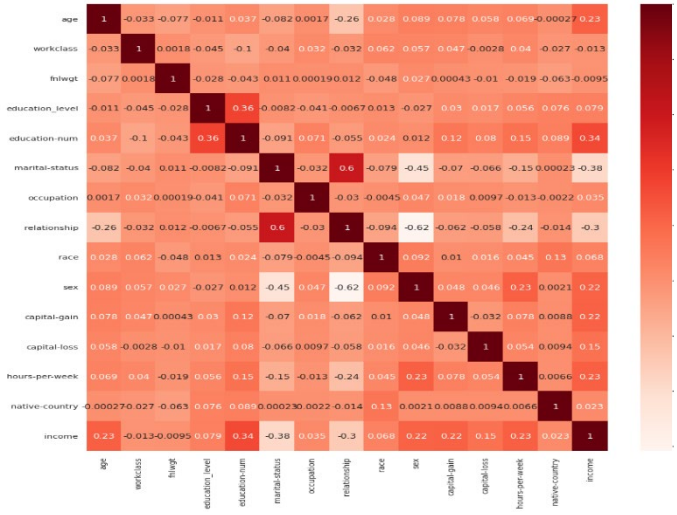
We use the describe method to check for the description of the dataframe. The describe() method returns description of the data in the Data Frame. If the Data Frame contains numerical data, the description contains this information for each column: count - The number of not-empty values. mean - The average (mean) value.

Now, to check the first 5 rows of the data frame we use df.head()

We draw a correlation matrix to check the correlation between each pair of variables. This helps us in further analyzing which features are important, and which are not.

From the correlation heatmap, we can see that the dependent feature 'income' is highly correlated with age, numbers of years of education, capital gain, and the number of hours per week.

From the correlation matrix, we observe that income has 34% correlation with 'Education_num', 23% correlation with 'hours_per_week' and 'age', and 22% correlation with 'Capital_gain'. The correlations are moderate.



Steps Involved in Data Preprocessing

2) **Cleaning Data**

We have the missing Values in our data set for the columns work class, occupation and native_country. So instead of removing the rows with missing values we are going to replace them, with the mean/mode. Hence, we replace '?' is 'Workclass' column by 'Private', 'Occupation' column by 'Prof-speciality' and 'Native_country' by 'United_States'.

We logically combined the data to reduce the number of categories in the features. We can keep Never-worked and Without-pay into one category. Also Group State-gov, Local-gov into Government. Add Self-emp-not-inc into 'Private' category We combine all the columns relevant to schools in 'School' category, Let's keep 'Doctorate' and 'Prof school' in a single category 'Doctorate', 'Assoc-acdm' and 'Assoc-voc' in one category 'Assoc', and 'HS-Grad' and 'Some-college' in one category 'College'.

Marital status: We combine 'Divorced', 'Married-spouse-absent', 'Separated', 'Widowed' and 'Married-AF-Spouse' to one category and name it as 'No spouse'.

Relationship Column: We combine 'Not-in-family', 'Own-child', 'Unmarried' and 'Other-relative' columns to a single category looking at the distributions, and name is as 'Other'.

Race column :We combine the categories 'Amer-Indian-Eskimo' and 'Other' to 'Others' category, since they have similar distributions.

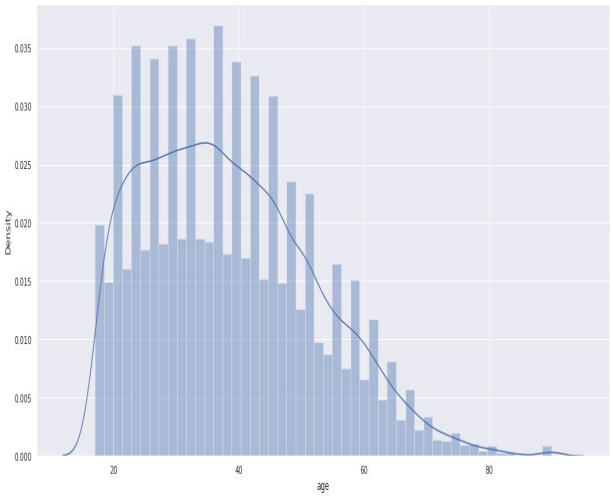
We used label encoder to normalize the labels and to transform categorical columns to numerical. Since majority of the classification models need input as 'int/float', and do not work on 'string' data, we encode our categorical columns using 'Label Encoder'

3) ***Scaling Data***

The next step is to bring the data to a common scale, since there are certain columns with very small values and some columns with high values. This process is important as values on a similar scale allow the model to learn better. We use standard scaler for this process .StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance.

4) ***Data Visualizations***

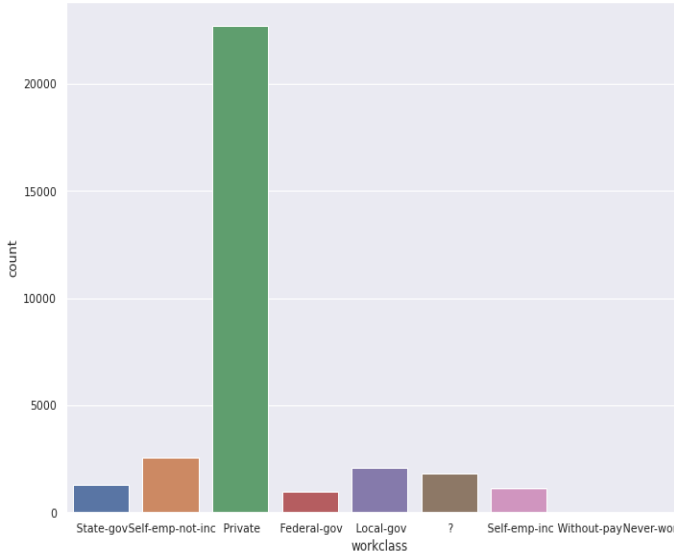
Age Column:



OBSERVATION**:**

We checked for null values in the age column and found that it has no null values in it. Distplot here is used to visualize the parametric distribution of the age variable. Our data has a right skewness, with most of the ages lying between 20 and 50. As people get older, the number continues to decrease. In this dataset, the greatest number of people are young, white, male, high school graduates with 9 to 10 years of education, and work 40 hours per week.

Workclass Column:



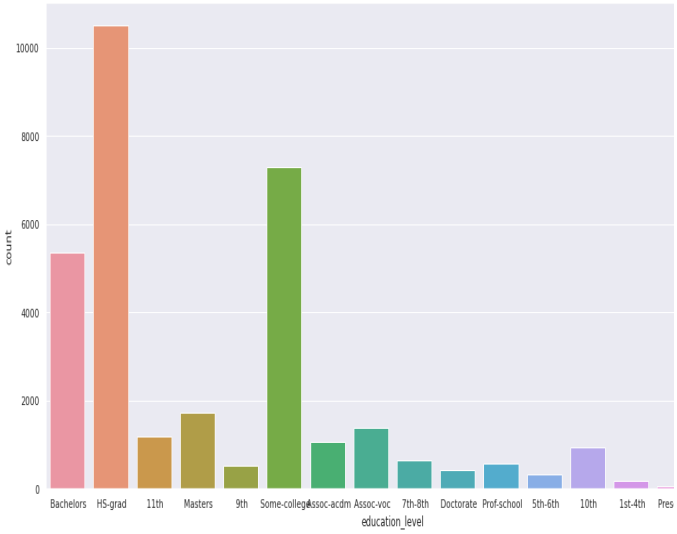
OBSERVATION:

We used value counts() function to get a series containing counts of unique values in work class column. We used seaborn. countplot() method to show the counts of observations in each categorical bin using bars.

When we look at the unique values for work class, we notice that there are seven different sorts of values, as well as some missing values denoted by a '?'. Null values account for 1836, or around 5% of the data. We see that most people work in the 'Private' sector.

We also notice something intriguing here: the values that are lacking 'Workclass' are also missing 'Occupation'.

Education Column:

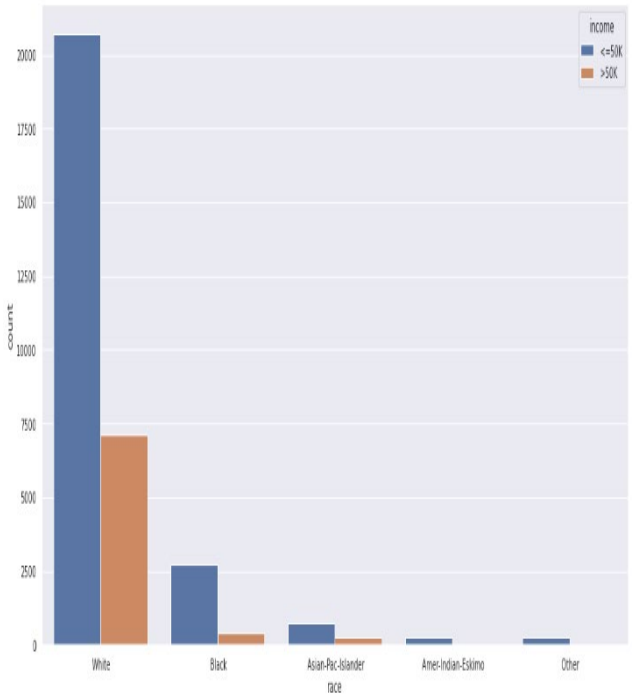


OBSERVATION:

We checked for its unique values and found that there are 16 different categories in the 'Education' column. Most of these categories fall within the 'School' category (different classes are divided into multiple categories).

There are no missing values in this column, and most persons have a 'HS-grad' education level, followed by 'Some-college' and 'Bachelors'.

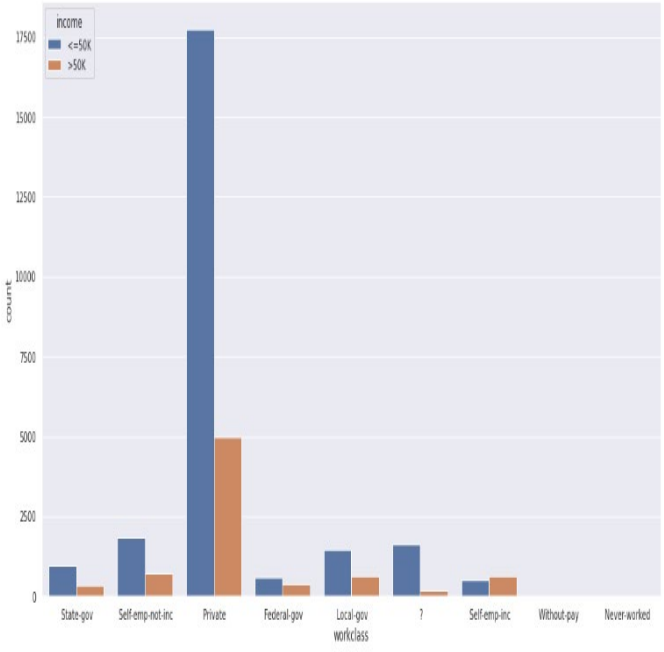
Race Vs Income:



OBSERVATION:

In this count plot, we realize that the Whites are large in number in the category of >50K while others being the least. Asians and Blacks have nearly equal population in this category whereas American Indian eskimos don't have any people in >50k income category.

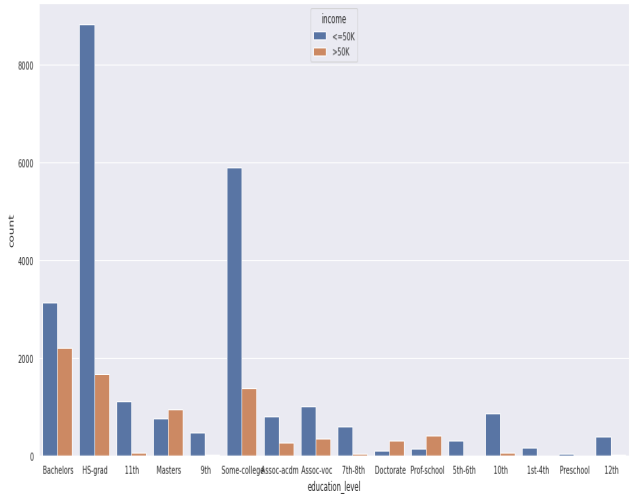
Work class Vs Income:



OBSERVATION:

We found that people of private sector have a lot of population >50K. Ratio of people earning more than 50K is higher in case Workclass is 'Self-emp-inc'.

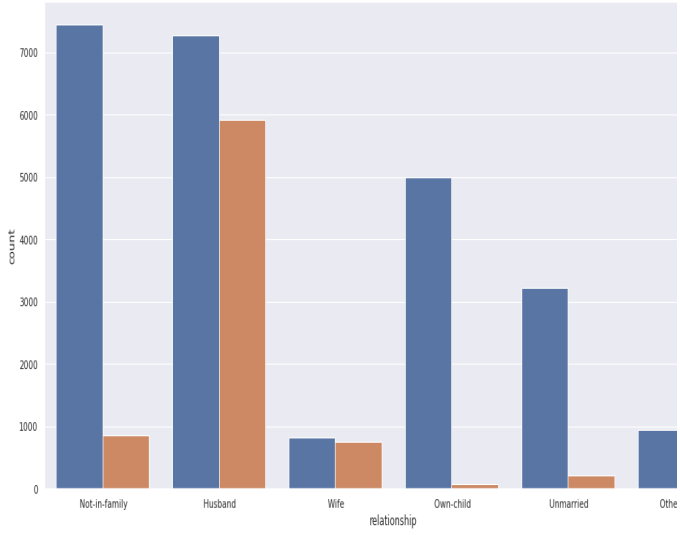
Education level vs Income:



OBSERVATION**:**

People with bachelor's degree have the highest income (>50K) and people with HS-grad have the highest income (<=50K). People with education level as 'Masters/Doctorate/Prof-school' have higher ratios of >50K earning, than <=50K. Bachelor's degree also has around 10:7 ratio of <=50K : >50K.

Relationship vs Income:



OBSERVATION**:**

Husband earns the most when compared to other relationships.

Based on the scatterplot of age, hours per week, and income, we can see that to make more than 50K(Dollars), a person must be at least 30 years old, or he/she should work at least 60 hours per week.

C. *Analysis from EDA*

If a person's race is 'White'/'Asian-pac-islander,' he or she has a good likelihood of earning more than 50K(Dollars). Males are more likely than females to earn more than 50K(Dollars).Number of People earning more than 50K is high in private sector when compared to Others. If Workclass is 'Self-emp-inc,' the percentage of those earning more than 50K(Dollars) is greater. People with a 'Masters/Doctorate/Prof-school' education have a larger ratio of >50K earning than those with a '=50K' education. The ratio of =50K: >50K for bachelor's degrees is roughly 10:7.

If the relationship status is 'Husband/Wife,' the odds of earning more than 50K(Dollars) are considerable. Based on the scatterplot of age, hours per week, and income, we can see that to make more than 50K(Dollars), a person must be at least 30 years old or he/she should work at least 60 hours per week.

D. *Feature Selection*

The curse of multicollinearity and the problem of overfitting can be solved by performing Feature Selection. The feature importance can be easily found by using the ExtraTreesClassifier.

We found that some features are least important and contribute nothing to prediction. As a result, we dropped all of them.

E. *Oversampling:*

The dependent feature 'Income' is highly imbalanced as 75.92% values have income less than 50k and 24.08% values have income more than 50k. This needs to be fixed as it results in a low F1 score. As we have a small dataset we can perform Oversampling using a technique like RandomOverSampler.

F. *Methods for fitting data into classification Models:*

The dataset is then split into X which contains all the independent features and Y which contains the dependent feature 'Income'.

We split our dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.

Here we are splitting the training and the testing data sets in the ratio of 80:20.

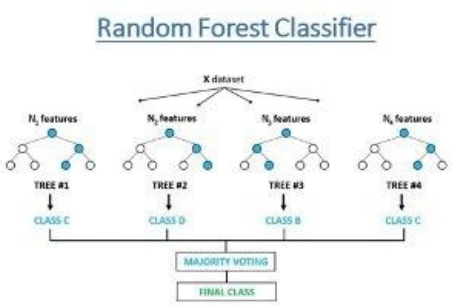
II. METHODS

A. Random Forest Classifier

Random forest is a Supervised learning algorithm that is used for both classification and regression. Since our project comes under classification, we thought this would be a great choice to start with. It is a type of bagging ensemble algorithm, which creates multiple decision trees simultaneously trying to learn from the dataset independent of one another. The final prediction is selected using majority voting.

Random forests are very flexible and give high accuracy as it overcomes the problem of overfitting by combining the results of multiple decision trees. Even for large datasets, random forests give a good performance. They also give good accuracy if our dataset has many missing values. But random forests are more complex and computationally intensive than

decision trees resulting in a time-consuming model building process. They are also harder to interpret and less intuitive than a decision tree.



This algorithm has some important parameters like max_depth, max_features, n_estimators, and min_sample_leaf. The number of trees which can be used to build the model is defined by n_estimators. Max_features determine the maximum number of features the random forest can use in an individual tree. The maximum depth of the decision trees is given by the parameter max_depth. The minimum number of samples required at a leaf node is given by min_sample_leaf.

Model Evaluation:

In this step, we will evaluate our model using accuracy_score as a metric. Accuracy is the ratio of correct predicted values over the total predicted values. It tells us how accurate our prediction is. We will tune the hyperparameters of our random forest classifier using RandomizedSearchCV which finds the best hyperparameters by searching randomly avoiding unnecessary computation. We will try to find the best values for 'n_estimators' and 'max_depth'.

Why Random Forest:

Random forests are created from subsets of data and the final output is based on the average or majority ranking and hence the problem of overfitting is taken care of. It is also worth noting that random forest classifiers are more stable and immune to curse of dimensionality - as in the case of our dataset, where we have 15 features. So we have chosen random forest as one of the methods to analyze this dataset.

This method gives an accuracy of **84.48** after tuning its hyperparameters. We have taken the classification report and it is as follows:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, Y_pred_ran_for))
```

	precision	recall	f1-score	support
<-50K	0.88	0.92	0.90	4908
>50K	0.71	0.62	0.66	1605
accuracy			0.84	6513
macro avg	0.80	0.77	0.78	6513
weighted avg	0.84	0.84	0.84	6513

B. Logistic Regression

Logistic regression is a classification algorithm, used when the value of the target variable is categorical in nature. Logistic regression is most commonly used when the data in question has binary output, so when it belongs to one class or another, or is either a 0 or 1.

Classification tasks have discrete categories, unlike regressions tasks. Here, by the idea of using a regression model to solve the classification problem, we rationally raise a question of whether we can draw a hypothesis function to fit to the binary dataset. The answer is that we will have to use a type of function, different from linear functions, called a logistic function, or a sigmoid function.

The sigmoid function/logistic function is a function that resembles an "S" shaped curve when plotted on a graph. It takes values between 0 and 1 and "squishes" them towards the margins at the top and bottom, labeling them as 0 or 1.The goal of classification with the dataset it to predict whether or not an income is likely to be greater than or less than or equal to 50k. There are only two categories this makes the classification task perfect for logistic regression.

Model Evaluation:

Now that we have prepped the data, we can feed it into the logistic regression classifier. First, we imported the logistic regression algorithm from Scikit-learn. Now we created an instance of the classifier and fit it to the training data. We created the predictions by running the classifier on the test dataset.

Finally, we checked to see how the classifier performed by importing some metrics and checking the predicted values against the actual values. This model is evaluated based on its accuracy score. The accuracy of this model is **83.86**. We have taken the classification and it is as follows.

	precision	recall	f1-score	support
0	0.86	0.94	0.90	4908
1	0.74	0.54	0.62	1605
accuracy			0.84	6513
macro avg	0.80	0.74	0.76	6513
weighted avg	0.83	0.84	0.83	6513

C. AdaBoost Classifier

Boosting refers to a class of machine learning ensemble algorithms where models are added sequentially and later models in the sequence correct the predictions made by earlier models in the sequence. AdaBoost, short for "Adaptive Boosting," is a boosting ensemble machine learning algorithm, and was one of the first successful boosting approaches.

We call the algorithm AdaBoost because, unlike previous algorithms, it adjusts adaptively to the errors of the weak hypotheses. AdaBoost combines the predictions from short one-level decision trees, called decision stumps, although other algorithms can also be used. Decision stump algorithms are used as the AdaBoost algorithm seeks to use many weak models and correct their predictions by adding additional weak models. The training algorithm involves starting with one decision tree, finding those examples in the training dataset that were misclassified, and adding more weight to those examples.

Another tree is trained on the same data, although now weighted by the misclassification errors. This process is

repeated until a desired number of trees are added. If a training data point is misclassified, the weight of that training data point is increased (boosted). A second classifier is built using the new weights, which are no longer equal. Again, misclassified training data have their weights boosted and the procedure is repeated.

The algorithm was developed for classification and involves combining the predictions made by all decision trees in the ensemble. A similar approach was also developed for regression problems where predictions are made by using the average of the decision trees. The contribution of each model to the ensemble prediction is weighted based on the performance of the model on the training dataset. We trained the model with an Ada boost classifier and predicted the outputs.

Model Evaluation:

Since the boosting algorithm involves combining of different weak learners, it results in "averaging" out the overfitting that is done in all the weak learners which usually leads to better accuracy. AdaBoosting also involves the process of assigning weightage based on prediction accuracy during training. By assigning higher weightage to mis predicted samples, AdaBoosting ensures that those learners are given more attention in the next training phases. It appears that when compared to Random Forest Classifier, the use of weighted decision stumps as opposed to decision trees has led to an increase in accuracy.

Random Forest Classifier follows a parallel ensemble approach aiming to reduce variance while AdaBooster follows a sequential approach aiming to reduce bias. Since the dataset is not a large dataset consisting of 32530 records, it can be observed that parallel processing doesn't result in significant change.

The accuracy of this model is **85.61**. We have taken the classification report and it is as follows:

```
print(classification_report(y_test, pred_ad))
```

	precision	recall	f1-score	support
0	0.88	0.94	0.91	4908
1	0.76	0.61	0.67	1605
accuracy			0.86	6513
macro avg	0.82	0.77	0.79	6513
weighted avg	0.85	0.86	0.85	6513

D. KNN

When KNN is used for classification, the output can be calculated as the class with the highest frequency from the K-most similar instances. Each instance in essence votes for their class and the class with the most votes is taken as the prediction. Class probabilities can be calculated as the normalized frequency of samples that belong to each class in the set of K most similar instances for a new data instance. For example, in a binary classification problem (class is 0 or 1):p(class=0) = count(class=0) / (count(class=0)+count(class=1)).

If we are using K and we have an even number of classes(2 in our case) it is a good idea to choose a K value with an odd number to avoid a tie. So, here we chose to take k as 5 initially. Ties can be broken consistently by expanding K by 1 and looking at the class of the next most similar instance in the training dataset.

Choosing best value of k: KNN reducing overfitting is a fact. On the other hand, there is a need to choose the best value for K. So now how do we choose K? Generally, we use the square root of the number of samples in the dataset as value for K. An optimal value must be found out since lower value may lead to overfitting and higher value may require high computational complication in distance. So, using an error plot may help. Another method is the elbow method. You can prefer to take root else can also follow the elbow method.

Let's dive deep into the different steps of K-NN for classifying a new data point:

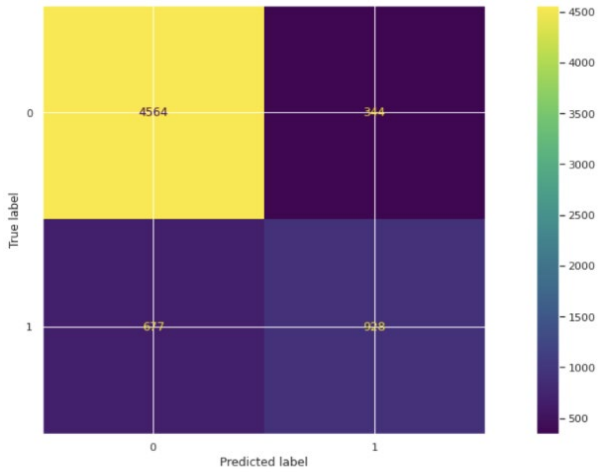
Step 1: Select the value of K neighbors(say k=5)
Step 2: Find the K (5) nearest data point for our new data point based on euclidean distance
Step 3: Among these K data points count the data points in each category
Step 4: Assign the new data point to the category that has the most neighbors of the new datapoint

Here we are using 1 parameter in the model creation. n_neighbors is setting as 5, which means 5 neighborhood points are required for classifying a given point. The distance metric we are using is Minkowski, the equation for it is given below

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

As per the equation, we have to select the p-value also. p = 1 , Manhattan Distance p = 2 , Euclidean Distance p = infinity , Cheybchev Distance In our problem, we are choosing the p as 2. Our Model is created, now we have to predict the output for the test set. This model is evaluated based on its accuracy score.

Confusion Matrix



Output

The accuracy of this model is 84.32.

Selecting best value for k

We plotted a graph of accuracy against the neighbors and found that for n=20, the accuracy is the highest which is around 84.32.

Why KNN?

In this dataset, we have a number of features on which the prediction is dependent on. This can be observed in the above mentioned correlation matrix. The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions. The quality of the predictions depends on the distance measure.

Predictions Analysis

From the confusion matrix, we have:

True Positives = 4564

False Positives(Type 1 Error) = 344

False Negatives(Type 2 Error) = 677

True Negatives = 928

Out of the 5492/6513 predictions are correct which gives makes the accuracy 84.32%. The rest 1021/6513 are the wrong predictions. We can calculate the precision and recall for this model from the data we got from the confusion matrix.

Precision = TP / (TP + FP) = 4564 / (4564 + 344) = 0.92

Recall = TP / (TP + FN) = 4564 / (4564 + 677) = 0.87

Ideally, we should have a high precision and recall. In our case, as the number of False Negatives are high, we don't have a good recall.

III. COMPARISONS

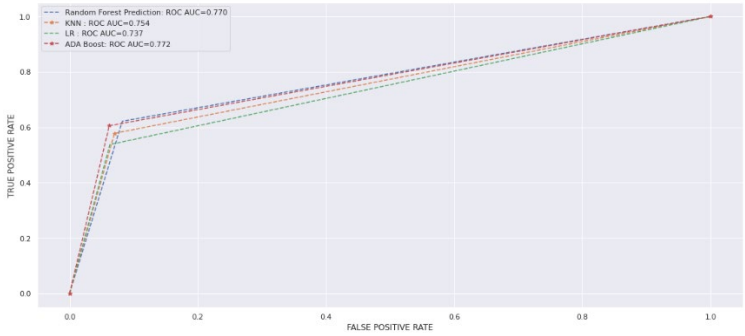
We have used the below metrics to compare our models.

Performance Metrics

1. Precision- The number of positive class predictions that actually, belong to the positive class is measured by precision.
2. Recall- The number of positive class predictions made out of all positive examples in the dataset is measured by recall.
3. F1-Score- F1-Score generates a single score that accounts for both precision and recall concerns in a single number.
4. Accuracy- It's the proportion of correct predictions to total input samples. It only works if each class has an equal number of samples.
5. AUC - The Area Under the Curve (AUC) is a curve that measures a classifier's ability to distinguish between classes.

The greater the AUC, the better.

MODEL	ACCURACY	PRECISION	RECALL	ROC
Random Forest Classifier	0.845	0.88,0.71	0.92,0.62	0.77
KNN	0.8432	0.87,0.73	0.93,0.58	0.754
Logistic Regression	0.838	0.86,0.74	0.94,0.54	0.737
Adaboost classifier	0.856	0.88,0.76	0.94,0.61	0.772



IV. CONCLUSION

In this analysis, we used four prominent classification techniques using US census data for Income. Adaboost has slightly better performed compared to random forest. These techniques can be used by charity organizations to classify potential donors.

V. REFERENCES

[1] Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.

[2] Avrim L. Blum and Pat Langley, Selection of relevant features and examples in machine learning, 0004–3702/97/1997 Elsevier Science B.V., Artificial Intelligence, vol. 97, pp. 245-271, 1997.

[3] Cover, Thomas M.; Hart, Peter E. (1967). "Nearest neighbor pattern classification".

[4] Ciyou Zhu, Richard Byrd, Jorge Nocedal and Jose Luis Morales. http://users.iems.northwestern.edu/~nocedal/lbfgsb.html.

[5] Y. Freund, R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.

[6] A.B. Gumelar, "An Anatomy of Machine Learning Data Visualization," 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), 2019, pp. 1-6, doi: 10.1109/ISEMANTIC.2019.8884340.