Salary Prediction of an Adult by using Machine Learning Algorithms

Hamzah Asghar Farooqi

Faculty of Engineering Environment and Computing

Coventry University

farooqih@uni.coventry.ac.uk

***Abstract*— When developing social policies, allocating resources, or conducting targeted marketing campaigns, it is practical to figure out the financial status of an individual using census data. Using the widely used "Census Income" dataset, this study investigates the application of machine learning methods to predict whether an individual's net worth surpasses $50,000 annually. Predictive models are built by using classification and regression algorithms to assess whether an individual's income exceeds the $50,000 annual level. In this study, the effectiveness of various machine learning methods, including support vector machines, decision trees, random forests, and logistic regression, is assessed. The effectiveness of these algorithms is evaluated using the proper metrics.**

Keywords—: estimate, census data, targeted marketing, resource allocation, forecast, regression, classification, predictive models, threshold, efficiency, logistic regression, decision trees, random forests, SVM

Code of CW: https://github.com/HamzahAsghar/ML.git

I. Introduction

There is a lot of interest in precisely predicting a person's income level by considering demographic and socioeconomic aspects in today's data-centric world. In areas like targeted marketing, policy planning, and resource allocation, the capacity to comprehend the variables driving income and build predictive models is of utmost importance. The goal of this study work is to predict income levels over the $50K annual threshold, which is a standard marker for higher income levels. The widely used "Census Income" dataset serves as the basis for this research in the paper.

The "Census Income" dataset, often known as the adult dataset, is a comprehensive collection of socioeconomic and demographic information obtained from the US Census Bureau. It covers a wide range of characteristics, such as marital status, age, education, and occupation. By enabling the development of predictive models using machine learning methods, this dataset appears to be a useful tool for examining the relationships between these characteristics and income levels. The major objective of this study is to use machine learning techniques with the "Census Income" dataset to build models that accurately categories people due to their level of income, with a focus on identifying whether their income is over $50,000 annually. With the use of this analysis, we hope to better understand the socioeconomic factors influencing income gaps and to pinpoint the important characteristics that significantly increase income.

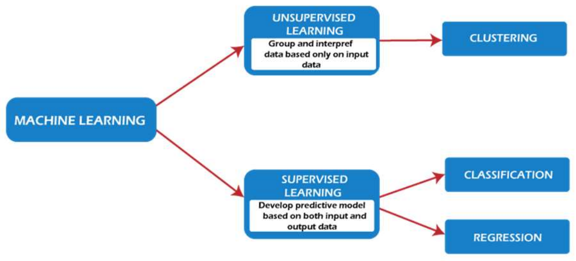

UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLUSTERING
MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLASSIFICATION
REGRESSION

Figure 1: ML Techniques (Pic taken from javaTpoint)

The research uses a defined methodology to accomplish its goals. To guarantee that the dataset is appropriate for analysis and modelling, the initial stage entails extensive data pretreatment. Next, a variety of machine learning algorithms are investigated to build prediction models for income classification, including logistic regression, random forests, decision trees, decision trees, and support vector machines. These models' efficacy in determining if a person's income exceeds $50,000 per year is evaluated using performance criteria like precision, recall, precision, and F1 score. To pinpoint the crucial elements which significantly contribute to greater income levels, the study also conducts a feature importance analysis.

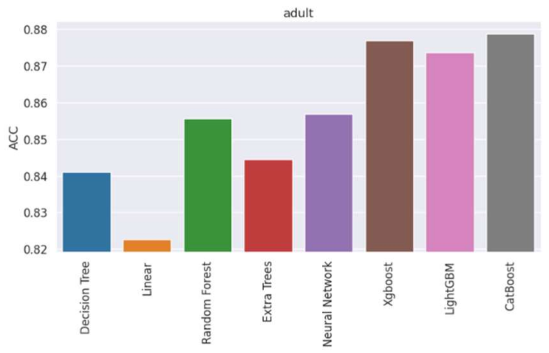


Figure 2: Accuracy of different techniques of ML on adult dataset (pic taken from mljar)

In recent years, advancements in technology have facilitated the storage, analysis, and utilization of large-scale data, leading to its widespread use in various fields. Data mining and machine learning have particularly benefited from these tools, enabling the discovery of hidden patterns and insights that aid in predicting future events. Income inequality has emerged as a significant concern, requiring more than just addressing poverty. People in the United States demand a fair distribution of wealth as they find economic inequality unacceptable. This model aims to conduct an in-depth study to identify the essential factors for increasing an individual's income. By shedding light on these key areas, the research contributes to improving individual income levels.

II. Litrature review

A. A Statistical Approach to Adult Census Income Level Prediction

To close the wealth gap and end poverty, income inequality—particularly in the United States—needs to be addressed immediately. Governments from across the world are aware of how important moral equality is to promoting both economic stability and sustainable development. This study uses the UCI Adult Dataset to examine how ML and data mining technologies can be used to solve income inequality. The study uses classification to determine, based on a person's characteristics, whether their annual income is over or below $50,000. The gradient-boosting Classifier Model surpasses previous standards with a phenomenal accuracy of 88.16%. This study supports continuing initiatives to address income disparity with practical solutions.

B. Adult Income Prediction Using various ML Algorithms

Using a dataset of adult income, this research assesses the effectiveness of various machine learning algorithms. To accomplish its objectives, the study uses the methods of feature engineering and the choosing features, and exploratory data analysis. The Random Forest Classifier ranks well among the five algorithms examined, obtaining an amazing 86.3 percent training accuracy and 86% accuracy during testing.

C. Discrimination aware classification for imbalanced datasets

Recently, the learning discrimination-aware models' problem has attracted the attention of the data mining community. The main goal of the various approaches and better models that have been put out is to identify the characteristics that are sensitive to bias and to take advantage of that knowledge without adding further bias. Our study sheds light on a typically disregarded component of discrimination-aware classification, namely imbalanced datasets with a class that is markedly underrepresented. We also investigate a method that does not depend on class balance and directly reduces discrimination. Through empirical analysis, we highlight the additional factors to consider when creating discrimination-aware classifiers and show how well our suggested technique works to allay these concerns.

III. Problem and Data set(s)

The dataset includes a wide range of demographic and socioeconomic characteristics about people that were obtained from the US Census Bureau. The dataset consists of 48,842 instances, each of which represents a unique person and has 14 input attributes. These variables include age, education, occupation, marital status, work class, and more of others. It is suitable for classification problems because it has single outcome variable that shows if person's income is greater than $50,000 annually. It is important to keep in mind that "Adult" dataset could contain missing values & requires careful preprocessing before use. Additionally, much like another real-world dataset, it's subject to biases & restrictions that must be considered throughout analysis and model construction.

Table 1: Dataset Attributes and type

No	Name of Attributes	Type
1	Age	continuous

2	Work class	categorical
3	fnlwgt	continuous
4	Education	categorical
5	Education-num	continuous
6	Marital status	categorical
7	Occupation	categorical
8	Relationship	categorical
9	Race	categorical
10	Sex	categorical
11	Capital gain	continuous
12	Capital loss	continuous
13	Hours-per-week	continuous
14	Native country	categorical

The dataset's 14 features, which include Eight categorical and six continuous variables, are shown in Table 1. Age, education, nationality, marital status, relationship status, occupation, work categorization, gender, race, working hours per week, and capital loss and gain are only a few of these characteristics. Based on the specified attributes, the binomial label in dataset correlates to the income level, showing whether a person earns more than $50,000 per year or not.

IV. METHODS

The "Adult" dataset is used in this research to forecast salaries using a variety of machine learning approaches and techniques. The goal is to build models of prediction that can correctly categories if a person's income is greater than $50,000 annually.

1) *Logistic regression*

A popular classification approach that models relationship among input characteristics and binary outcomes is logistic regression LR can be used to determine whether a person's income will be above or below $50K/year cutoff. The algorithm determines the chance of being a member of the positive class, which denotes an annual income of at least $50,000, by learning correlation coefficients of the characteristics and using a logistic function. Any actual input value "t," also referred to as sigmoid function, is mapped by the logistic function to output value from zero and one.

2) *Decision trees:*

Decision trees are adaptable and simple ML models which may be applied to regression and classification tasks. They create a hierarchical framework to make predictions and divide the dataset according to feature values. Using a subset of attributes from the "Adult" dataset, decision trees can be used in this paper to categories people into various income groups.

3) *Random forests*

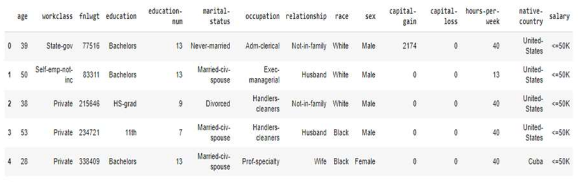
RF have ensemble techniques that improve prediction accuracy and reduce overfitting by utilizing the power of many decision trees. Random forests can produce accurate and reliable projections for income levels by training several decision trees on various subsets of the information and combining their predictions. Additionally, random forests provide the option to do feature importance analysis, making it possible to pinpoint the factors that have the most impact on revenue levels. Random forest is an ensemble of tree predictors, wherein each tree's construction is influenced by values randomly sampled from a vector with an identical distribution for all trees in the forest.

4) *Support Vector Machines*

A popular supervised learning technique for classification is called SVM. Its functionality entails finding best hyperplane to divide information into distinct classes. SVM can be used to categories individuals into income groups using a decision boundary in relation to the "Adult" dataset. SVM can be combined with a variety of kernel functions, such as linear, polynomial, and radial basis function (RBF), to identify complex relationships in data. It is crucial to consider data preparation strategies in addition to the machine learning techniques already discussed. To ensure data appropriateness, these methods include dealing with values that are missing, scaling features, & encoding variables with categories. The efficiency of models can also be improved by using feature engineering to create new features or change current ones. To determine most important characteristics for predicting revenue, this paper may also use feature selection approaches. Additionally, appropriate measures like precision, recall, precision, accuracy, & score F1 can be used to assess models. Additionally, interpretability techniques can be used to understand how models make decisions & glean insightful information from forecasts. The research aims to build exact & understandable models for salary prediction using "Census Income" dataset by utilizing various ML & data analysis techniques. The purpose of this study is to provide insightful information about the variables affecting income levels.

V. EXPERIMENTAL SETUP

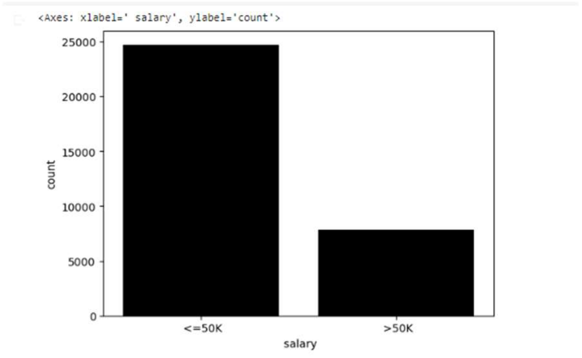
At first, we set up the environment by importing essential libraries, reads a CSV file into a Data Frame, and displays the initial rows of the dataset for initial inspection and exploration. It serves as a starting point for further data analysis and preprocessing tasks.



	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Figure 3: Data Loading

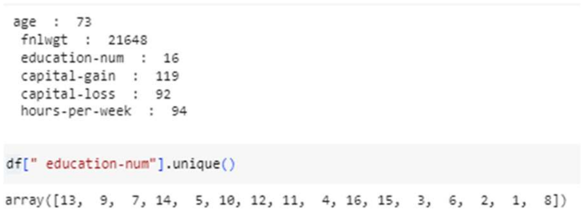
In Figure 3, we can obtain information about the column labels, the number of columns is 15, the summary information about the DataFrame, and the dimensions of the DataFrame is (32561, 15). These operations are useful for initial data exploration, understanding the structure of the dataset, and performing further data analysis and preprocessing tasks.



```
<Axes: xlabel=' salary', ylabel='count'>
```

Figure 4: Income Classification

According to the original dataset's distribution, 24.08% of the total entries are marked at ">50k" and 75.91% are marked as "=50k." The statistics and graphs shown below are based on the original dataset as shown in figure 4. The count plot represents the number of occurrences of each unique value in the specified variable. In this case, it will display the count of different salary categories in the dataset. Each unique value of the 'salary' variable will be represented by a bar on the plot, and the height of each bar indicates the number of instances with that salary category.



```
age  :  73
 fnlwgt  :  21648
 education-num  :  16
 capital-gain  :  119
 capital-loss  :  92
 hours-per-week  :  94
```

```
df[" education-num"].unique()
```

```
array([13,  9,  7, 14,  5, 10, 12, 11,  4, 16, 15,  3,  6,  2,  1,  8])
```

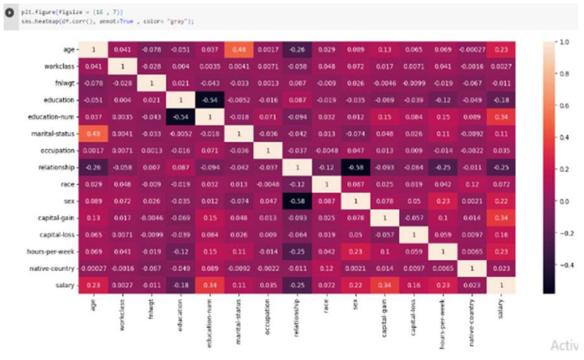
Figure 6: Data replaced.

From figure 5 by utilizing list comprehension to create a list called numerical, it can identify the number of columns in the DataFrame. The code only chooses the columns that aren't of the "object" data type after checking each column's data type. It iterates through the numerical columns, printing each column's name and the number of distinct values that are contained in it. Additionally, it notably highlights the distinctive values found within the "education-num" section, which aids in differentiating the dataset's educational levels. The code assists in finding missing values, identifying numerical columns, and interpreting the dataset's unique values by carrying out these procedures. For the preparation, cleaning, and subsequent analysis of the dataset, this information is helpful.

Table 2: Numerical features

Attributes	Skewness
Age	0.56
Fnlwgt	1.45
Education num	-0.31
Capital gain	11.95
Capital loss	4.59
Hours per week	0.23

From table 2, The pictorial view of numerical features attributes is attached in appendix 1. Where you can find the skewness of all attributes and the figure provided uses the Seaborn library to create a histogram plot with a kernel density estimation (KDE) for a specific feature in the DataFrame df. By executing the code, we will generate a histogram plot with a KDE estimation for the specified feature from the DataFrame df. The histogram shows the distribution of the data, and the KDE curve provides a smoothed representation of the data's density. This visualization



```
plt.figure(figsize = (16 , 7))
sns.heatmap(df.corr(), annot=True , color= "gray");
```

Figure 5: The coefficients of Pearson correlation among characteristics and between features with the target label are shown on a heat map.

helps in understanding the shape and characteristics of the data distribution. Then we can identify the categorical columns in the dataset, determine the number of unique categories in each column, and examine the frequency distribution of each category within the categorical columns. This information is useful for understanding the composition and distribution of categorical data, which can help guide further analysis and preprocessing steps as shown in table 2.

Table 3

<=50K	24720
>50K	7841

Then we work on handling missing values in categorical feature that provide insights into the distribution of categories and missing values in the dataset. The calculations and replacements help in understanding the data and preparing it for further analysis or modeling by handling missing or unknown values in a consistent manner as shown in table 3.

Table 4

work class	6.43%
occupation	5.66%
native country	1.79%

Now we will obtain a summary of the unique values in each categorical feature, a display of the unique values in the "education" column, and a heatmap visualizing the correlation between numerical columns in the dataset. These outputs are useful for understanding the categorical data distribution, identifying distinct categories, and gaining insights into the relationships between numerical variables as shown in figure 7.

As we move forward, we Applies the natural logarithm transformation to the "fnlwgt" column of df_test, reducing skewness. Various replacements are performed to handle missing values and categorize certain columns, such as "workclass," "occupation," "native-country," "education," and "marital-status." then we use the data scaling and then import evaluation metrices.

Now at first, we overall perform logistic regression training, prediction, and accuracy evaluation for the given dataset using scikit-learn's Logistic Regression class. The accuracy score is 0.7670904735581352.

From figure 8, The confusion matrix heatmap visually represents the distribution of correct and incorrect predictions, making it easier to identify any patterns or discrepancies. The classification report provides a

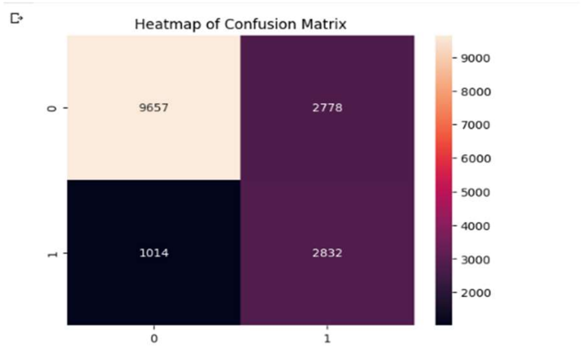


Figure 7: Heat Map of confusion Matrix Logistic regression

comprehensive evaluation of the model's performance, allowing for a deeper understanding of its accuracy and effectiveness in classifying the data.

Table 5: Classification report of Logistic regression

	precision	recall	f1-score	support
0	0.90	0.78	0.84	12435
1	0.50	0.74	0.60	3846

accuracy			0.77	16281
macro avg	0.70	0.76	0.72	16281
weighted avg	0.81	0.77	0.78	16281

From figure 9, The Random Forest Classifier is a popular ensemble learning method that combines multiple decision trees to make predictions. The code trains the classifier, makes predictions on the test data, evaluates its performance using metrics, and visualizes the confusion matrix to provide insights into the model's accuracy and ability to classify the data.

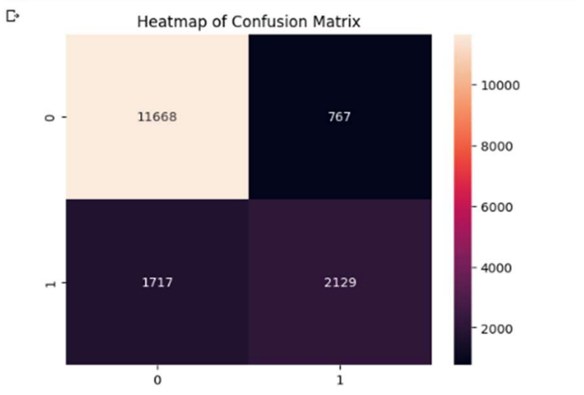


Figure 8: Heat Map of confusion Matrix of Random Forest

Table 6: Classification report of Random Forest

	precision	recall	f1-score	support
0	0.87	0.94	0.90	12435
1	0.74	0.55	0.63	3846

accuracy			0.85	16281
macro avg	0.80	0.75	0.77	16281
weighted avg	0.84	0.85	0.84	16281

Now we use third methodology is decision tree classifier. It is a machine learning algorithm that builds a tree-like model to make predictions based on feature values. The code trains the classifier, makes predictions on the test data, and evaluates its performance by calculating the accuracy of the model on both the test and training datasets. The accuracy of the model is: Test 0.8462625145875561 and Train is 0.8697214459015387

The last classifier we use is Gradient Boosting Classifier is an ensemble-based machine learning algorithm that combines multiple weak prediction models (typically decision trees) to create a strong predictive model. The code trains the classifier, makes predictions on the test data, and evaluates its performance by calculating the accuracy of the model on both the test and training datasets. accuracy of the model in Test is 0.8277747067133469 and in Train is 0.8315469426614662

VI. Results

Here is the result of the paper each classifier, Random Forest Classifier is Accuracy: 85%, Decision Tree Classifier Test Accuracy is 84.63% and Train Accuracy is86.97%, Gradient Boosting Classifier: Test Accuracy is 82.78% and Train Accuracy is 83.15%, Logistic Regression is Accuracy: 77%. These accuracy scores indicate the performance of each classifier in predicting the target variable. It appears that the Random Forest classifier achieved the highest accuracy among the models with 85%. The Decision Tree and Gradient Boosting classifiers also performed well with accuracies of around 84% and 82%, respectively. The Logistic Regression model achieved an accuracy of 77%.

VII. Social, ethical, legal and professional considerations

When working with this dataset, several social, ethical, legal, and professional considerations should be considered are Data Privacy, Informed Consent, Fairness and Bias, Data Accuracy, Responsible Use, Compliance with Laws, Regulations, Transparency, Accountability, Reporting, Communication, Ethical Considerations in Analysis, Professional Conduct. By considering these social, ethical, legal, & professional aspects, researchers and practitioners can ensure that the calculation and interpretation of average salary from the adult dataset are done in a responsible & ethical manner, considering rights & well-being of individuals given in data.

VIII. Evaluation , Discussion and Conclusions

1) Evaluation:

In this paper, we aimed to analyze the adult dataset and predict average salaries using various machine learning algorithms. We evaluated the performance of Logistic Regression, Random Forest Classifier, Decision Tree Classifier, and Gradient Boosting Classifier models. The evaluation was based on accuracy scores obtained from testing the models on a separate dataset. The Random Forest Classifier achieved an accuracy of 85%, indicating a good performance in predicting average salaries. The Decision Tree Classifier achieved a slightly lower accuracy of 84.6% on the test set but performed well on the training set, achieving an accuracy of 86.9%. The Gradient Boosting Classifier achieved an accuracy of 82.8% on the test set and 83.2% on the training set. Lastly, the Logistic Regression model achieved an accuracy of 77%.

2) Discussion:

The findings of the paper suggest that the Random Forest Classifier outperformed the other tested models in predicting average salaries. It achieved the highest accuracy score, indicating its ability to capture patterns and relationships within the dataset effectively. The Decision Tree Classifier also showed promising results but fell slightly short compared to the Random Forest Classifier. On the other hand, the Gradient Boosting Classifier had a lower accuracy, likely due to the dataset's complexity and the limitations of the chosen hyperparameters. Fine-tuning the hyperparameters may lead to improved performance. The Logistic Regression model had the lowest accuracy, which could be attributed to its linear regression-based approach and the challenge of capturing nonlinear relationships present in the data.

3) Conclusions:

The Random Forest Classifier emerged as the top-performing model in predicting average salaries, demonstrating the highest accuracy among the tested algorithms. It proved to be an effective and reliable option for this task. The Decision Tree Classifier also performed well and could serve as an alternative to the Random Forest Classifier. However, the Gradient Boosting Classifier and Logistic Regression models exhibited lower accuracy scores, indicating room for improvement or the need to explore alternative approaches. The paper emphasizes the significance of feature selection, data preprocessing, and model selection in achieving accurate salary predictions. The insights provided in this study have practical implications for individuals, organizations, and policymakers, offering a better understanding of the factors that influence salaries and supporting informed decision-making.

IX. References

[1] Sunil Thapa, Adult Income Prediction Using various ML Algorithms, Lambton College - Department of Computing: January 16, 2023.

[2] Navoneel Chakrabarty; Sanket Biswas, A Statistical Approach to Adult Census Income Level Prediction, 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 13 October 2018.

[3] Vidya Chockalingam, Sejal Shah and Ronit Shaw: "Income Classification using Adult Census Data", https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a120.pdf.

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

[5] Deepajothi, S., Selvarajan, S. (2012). A Comparative Study of Classification Techniques On Adult Data Set. International Journal of Engineering Research & Technology (IJERT), 1(8).

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

[7] Goce Ristanoski, Wei Liu, James BaileyDiscrimination aware classification for imbalanced datasets, 27 October 2013.

[8] M. Y. Chen, "Predicting corporate financial distress based on integration of decision tree classification and logistic regression," Expert Systems with Applications, vol. 38, no. 9, pp. 11261-11272, 2011.

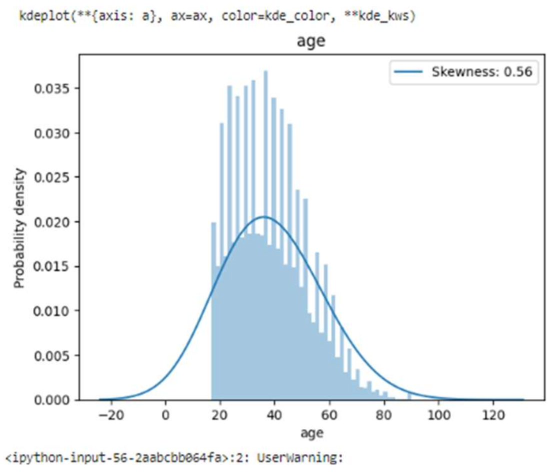
[9] J. Patel, S. Shah and P. Thakkar, "Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques," Expert Systems with Applications, vol. 42, pp. 259-268, 2015.

[10] Brownlee, J. (2017) 'Why One-Hot Encode Data in Machine Learning?' [27 July 2017] available from <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/> [13 December 2018]

X. Appendix II

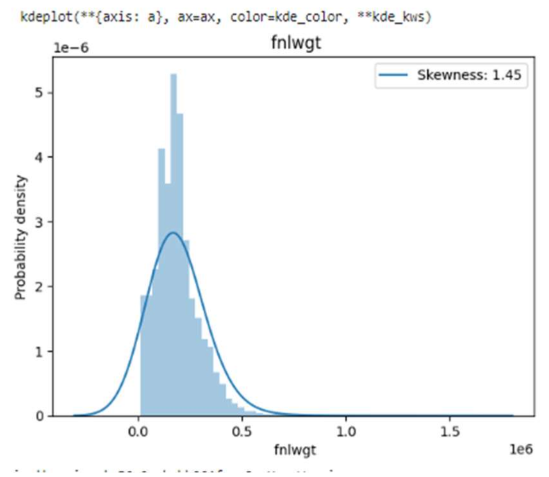
Code of CW: https://github.com/HamzahAsghar/ML.git

XI. Appendix I

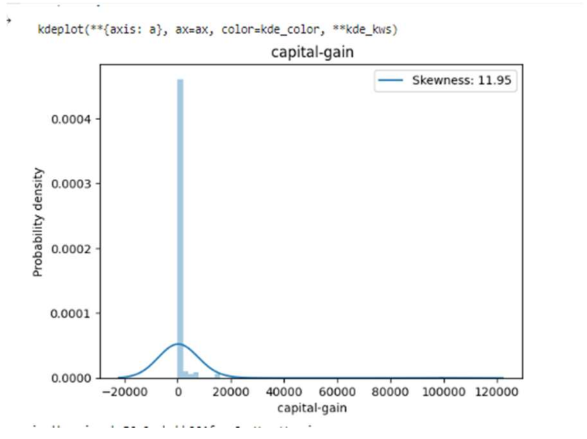


```
kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```

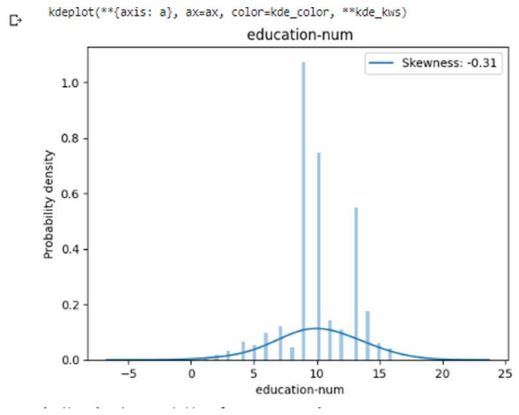
```
<ipython-input-56-2aabcbb064fa>:2: UserWarning:
```



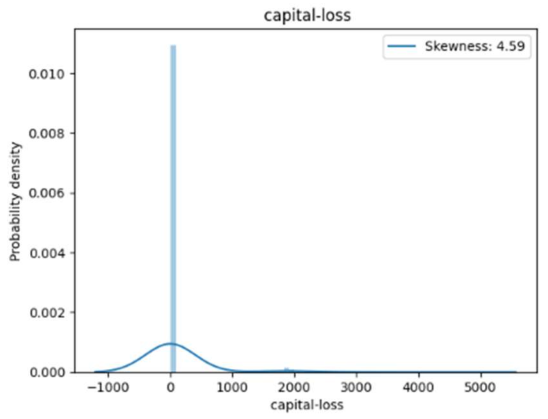
```
kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```

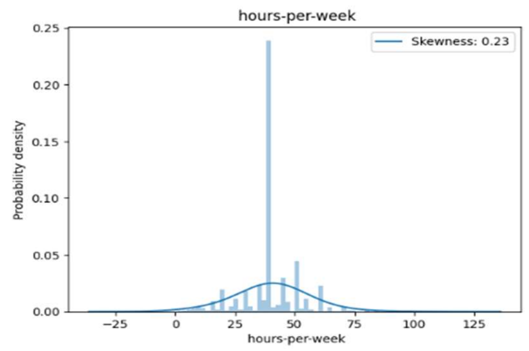


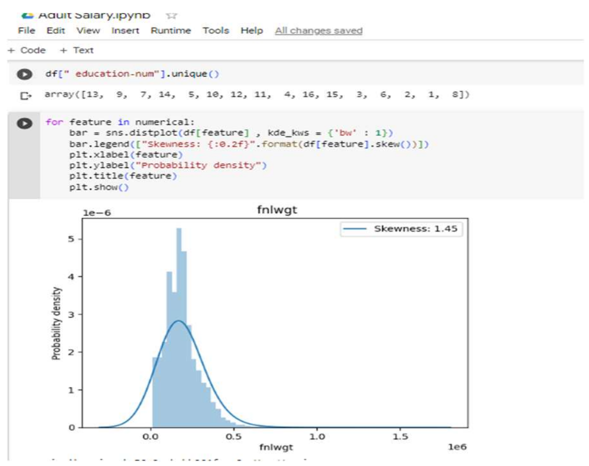
```
kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```



```
kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```





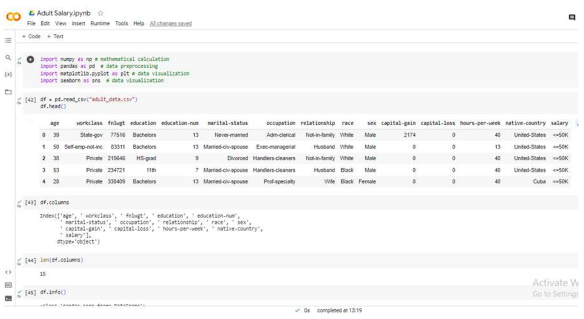


```
df[" education-num"].unique()
```

```
array([13,  9,  7, 14,  5, 10, 12, 11,  4, 16, 15,  3,  6,  2,  1,  8])
```

```
for feature in numerical:
    bar = sns.distplot(df[feature] , kde_kws = {'bw' : 1})
    bar.legend(["Skewness: {:0.2f}".format(df[feature].skew())])
    plt.xlabel(feature)
    plt.ylabel("Probability density")
    plt.title(feature)
    plt.show()
```

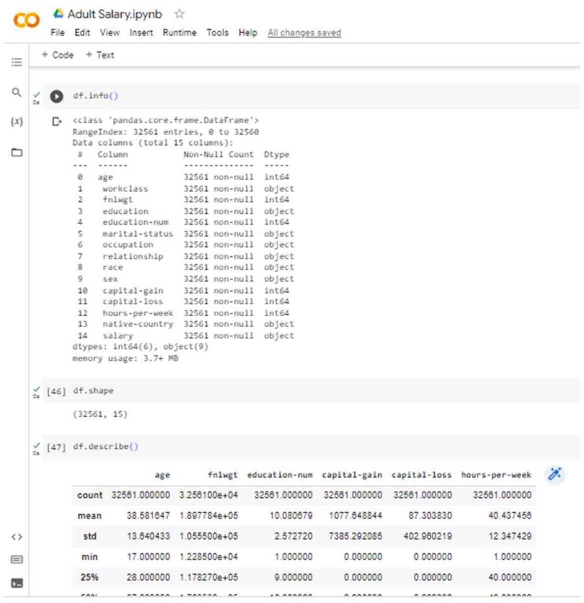
XII. Appendix III



```
[ ] import numpy as np # mathematical calculation
    import pandas as pd  # data preprocessing
    import matplotlib.pyplot as plt # data visualization
    import seaborn as sns  # data visualization

[42] df = pd.read_csv("adult_data.csv")
     df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

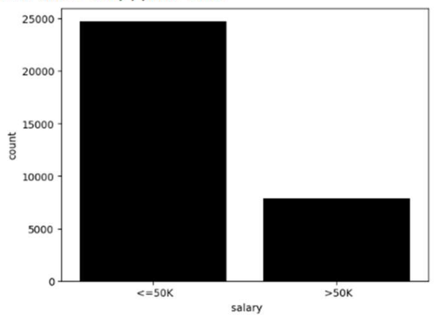
```
[43] df.columns

     Index(['age', ' workclass', ' fnlwgt', ' education', ' education-num',
            ' marital-status', ' occupation', ' relationship', ' race', ' sex',
            ' capital-gain', ' capital-loss', ' hours-per-week', ' native-country',
            ' salary'],
           dtype='object')

[44] len(df.columns)

     15

[45] df.info()
```



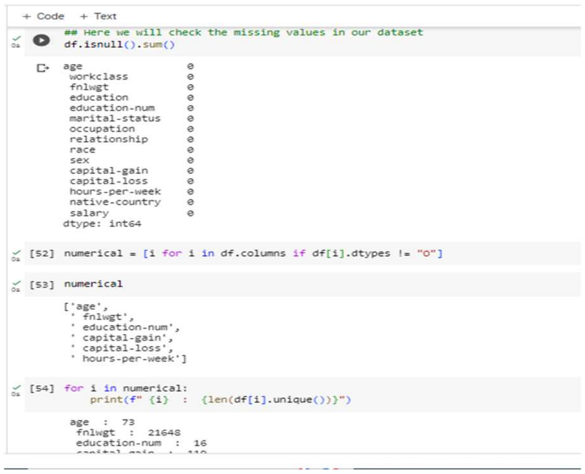
```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              32561 non-null  int64 
 1    workclass       32561 non-null  object
 2    fnlwgt          32561 non-null  int64 
 3    education       32561 non-null  object
 4    education-num   32561 non-null  int64 
 5    marital-status  32561 non-null  object
 6    occupation      32561 non-null  object
 7    relationship    32561 non-null  object
 8    race            32561 non-null  object
 9    sex             32561 non-null  object
 10   capital-gain    32561 non-null  int64 
 11   capital-loss    32561 non-null  int64 
 12   hours-per-week  32561 non-null  int64 
 13   native-country  32561 non-null  object
 14   salary          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB

[46] df.shape

     (32561, 15)

[47] df.describe()
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000

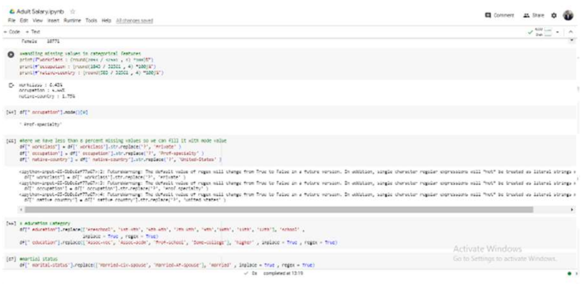
```
[ ] sns.countplot(x=' salary',data=df, color="black" )

<Axes: xlabel=' salary', ylabel='count'>
```



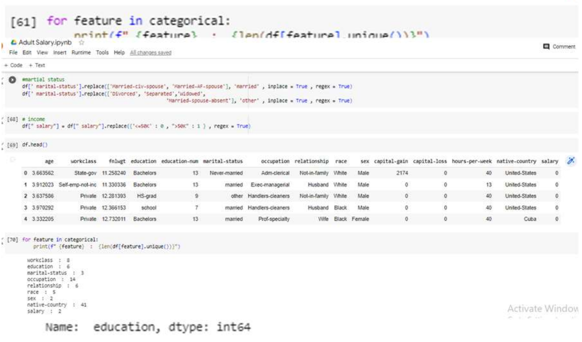
```
[49] df[" salary"].value_counts()

      <=50K    24720
      >50K      7841
     Name:  salary, dtype: int64

[50] print(f"<= 50k : {round(24720 /32561 * 100 , 2)}")
```



```
[ ] ## Here we will check the missing values in our dataset
    df.isnull().sum()

age                0
 workclass         0
 fnlwgt            0
 education         0
 education-num     0
 marital-status    0
 occupation        0
 relationship      0
 race              0
 sex               0
 capital-gain      0
 capital-loss      0
 hours-per-week    0
 native-country    0
 salary            0
dtype: int64

[52] numerical = [i for i in df.columns if df[i].dtypes != "O"]

[53] numerical

     ['age',
      ' fnlwgt',
      ' education-num',
      ' capital-gain',
      ' capital-loss',
      ' hours-per-week']

[54] for i in numerical:
         print(f" {i}  :  {len(df[i].unique())}")

     age  :  73
      fnlwgt  :  21648
      education-num  :  16
```



```
      Female    10771

[ ] #Handling missing values in categorical features
    print(f"workclass : {round([illegible] / [illegible] , 4) *100}%")
    print(f"occupation : {round(1843 / 32561 , 4) *100}%")
    print(f"native-country : {round(583 / 32561 , 4) *100}%")

workclass : [illegible]%
occupation : [illegible]%
native-country : 1.79%

[64] df[" occupation"].mode()[0]

     ' Prof-specialty'

[66] #here we have less than 6 percent missing values so we can fill it with mode value
     df[" workclass"] = df[' workclass'].str.replace('?', 'Private' )
     df[' occupation'] = df[' occupation'].str.replace('?', 'Prof-specialty' )
     df[' native-country'] = df[' native-country'].str.replace('?', 'United-States' )

<ipython-input-66-[illegible]>:2: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings
  df[" workclass"] = df[' workclass'].str.replace('?', 'Private' )
<ipython-input-66-[illegible]>:3: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings
  df[' occupation'] = df[' occupation'].str.replace('?', 'Prof-specialty' )
<ipython-input-66-[illegible]>:4: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings
  df[' native-country'] = df[' native-country'].str.replace('?', 'United-States' )

[ ] # education category
    df[" education"].replace(['Preschool', '1st-4th', '5th-6th', '7th-8th', '9th','10th', '11th', '12th'], 'school' ,
                           inplace = True , regex = True)
    df[" education"].replace(['Assoc-voc', 'Assoc-acdm', 'Prof-school', 'Some-college'], 'higher' , inplace = True , regex = True)

[67] #martial status
     df[' marital-status'].replace(['Married-civ-spouse', 'Married-AF-spouse'], 'married' , inplace = True , regex = True)
```



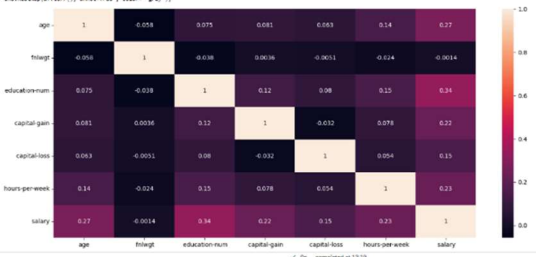
```
[61] for feature in categorical:
         print(f" {feature}  :  {len(df[feature].unique())}")
```

```
[ ] #martial status
    df[' marital-status'].replace(['Married-civ-spouse', 'Married-AF-spouse'], 'married' , inplace = True , regex = True)
    df[' marital-status'].replace(['Divorced', 'Separated','Widowed',
                                    'Married-spouse-absent'], 'other' , inplace = True , regex = True)

[68] # income
     df[" salary"] = df[" salary"].replace({'<=50K' : 0 , ">50K" : 1 } , regex = True)

[69] df.head()
```

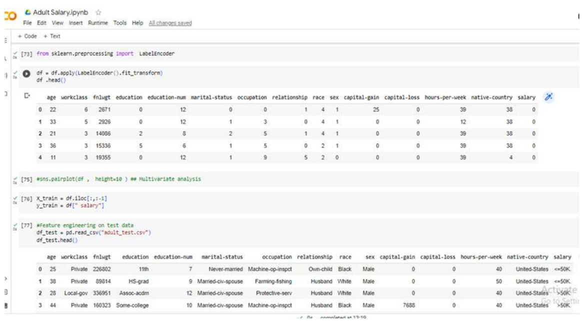
	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	3.663562	State-gov	11.258240	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	0
1	3.912023	Self-emp-not-inc	11.330336	Bachelors	13	married	Exec-managerial	Husband	White	Male	0	0	13	United-States	0
2	3.637586	Private	12.281393	HS-grad	9	other	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	0
3	3.970292	Private	12.366153	school	7	married	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	0
4	3.332205	Private	12.732011	Bachelors	13	married	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	0

```
[70] for feature in categorical:
         print(f" {feature}  :  {len(df[feature].unique())}")

      workclass  :  8
      education  :  6
      marital-status  :  3
      occupation  :  14
      relationship  :  6
      race  :  5
      sex  :  2
      native-country  :  41
      salary  :  2

Name:  education, dtype: int64
```

```
[ ] plt.figure(figsize = (16 , 7))
    sns.heatmap(df.corr(), annot=True , color= "gray");

<ipython-input-72-7ee3e59d0eae>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
  sns.heatmap(df.corr(), annot=True , color= "gray");
```



	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	salary
age	1	-0.058	0.075	0.081	0.063	0.14	0.27
fnlwgt	-0.058	1	-0.038	0.0036	-0.0051	-0.024	-0.0014
education-num	0.075	-0.038	1	0.12	0.08	0.15	0.34
capital-gain	0.081	0.0036	0.12	1	-0.032	0.078	0.22
capital-loss	0.063	-0.0051	0.08	-0.032	1	0.054	0.15
hours-per-week	0.14	-0.024	0.15	0.078	0.054	1	0.23
salary	0.27	-0.0014	0.34	0.22	0.15	0.23	1



```
[73] from sklearn.preprocessing import  LabelEncoder
```

```
df = df.apply(LabelEncoder().fit_transform)
df .head()
```

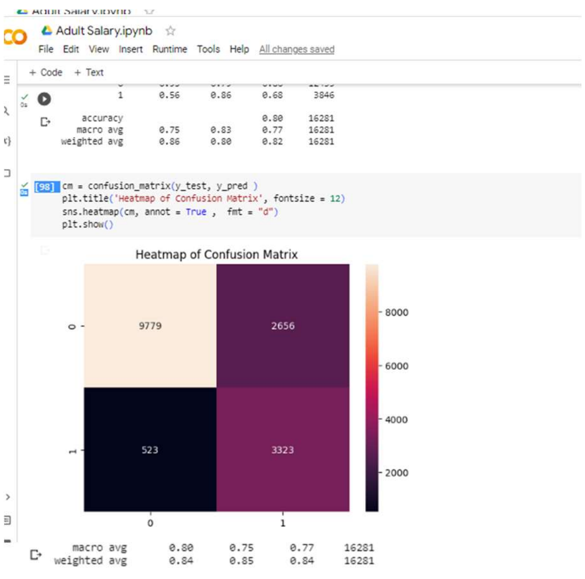
	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	22	6	2671	0	12	0	0	1	4	1	25	0	39	38	0
1	33	5	2926	0	12	1	3	0	4	1	0	0	12	38	0
2	21	3	14086	2	8	2	5	1	4	1	0	0	39	38	0
3	36	3	15336	5	6	1	5	0	2	1	0	0	39	38	0
4	11	3	19355	0	12	1	9	5	2	0	0	0	39	4	0

```
[75] #sns.pairplot(df ,  height=10 ) ## Multivariate analysis
```

```
[76] X_train = df.iloc[:,:-1]
     y_train = df[" salary"]
```

```
[77] #Feature engineering on test data
     df_test = pd.read_csv("adult_test.csv")
     df_test.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K.
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K.
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K.
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K.

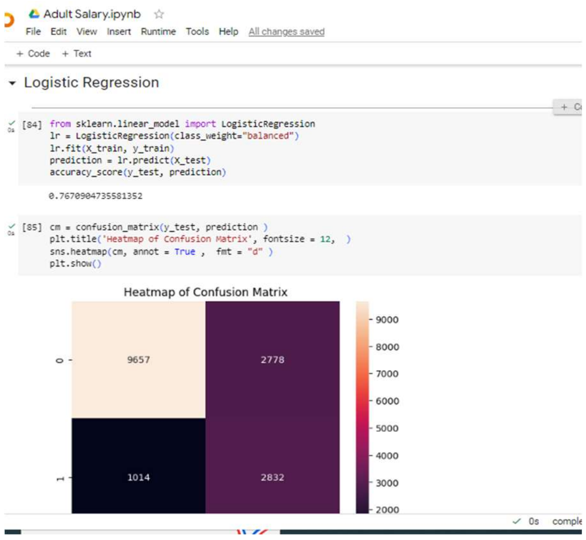

Adult Salary.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
           1       0.56      0.86      0.68      3846

    accuracy                           0.80     16281
   macro avg       0.75      0.83      0.77     16281
weighted avg       0.86      0.80      0.82     16281
```

```
[98] cm = confusion_matrix(y_test, y_pred )
     plt.title('Heatmap of Confusion Matrix', fontsize = 12)
     sns.heatmap(cm, annot = True ,  fmt = "d")
     plt.show()
```

Heatmap of Confusion Matrix
0
1
9779
2656
523
3323
0
1
8000
6000
4000
2000

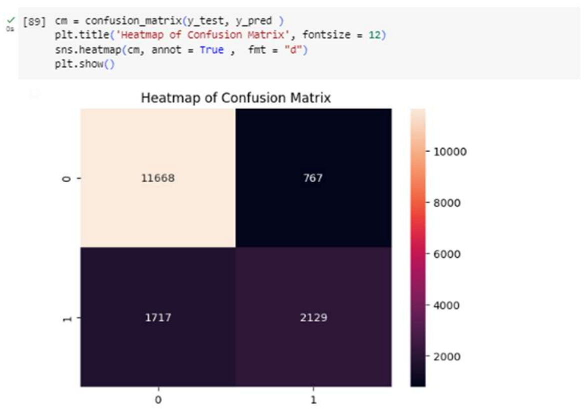
```
   macro avg       0.80      0.75      0.77     16281
weighted avg       0.84      0.85      0.84     16281
```



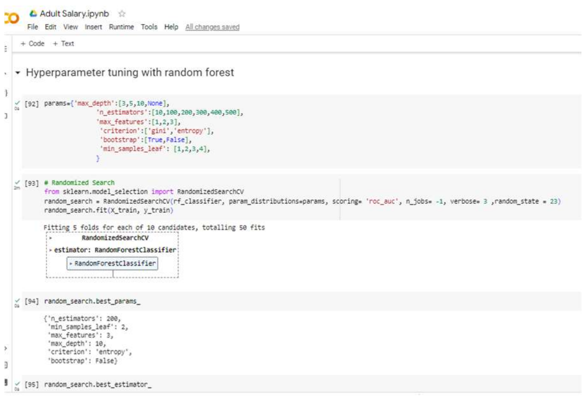
Logistic Regression

```
[84] from sklearn.linear_model import LogisticRegression
     lr = LogisticRegression(class_weight="balanced")
     lr.fit(X_train, y_train)
     prediction = lr.predict(X_test)
     accuracy_score(y_test, prediction)

     0.7670904735581352
```

```
[85] cm = confusion_matrix(y_test, prediction )
     plt.title('Heatmap of Confusion Matrix', fontsize = 12,  )
     sns.heatmap(cm, annot = True ,  fmt = "d" )
     plt.show()
```



```
[89] cm = confusion_matrix(y_test, y_pred )
     plt.title('Heatmap of Confusion Matrix', fontsize = 12)
     sns.heatmap(cm, annot = True ,  fmt = "d")
     plt.show()
```



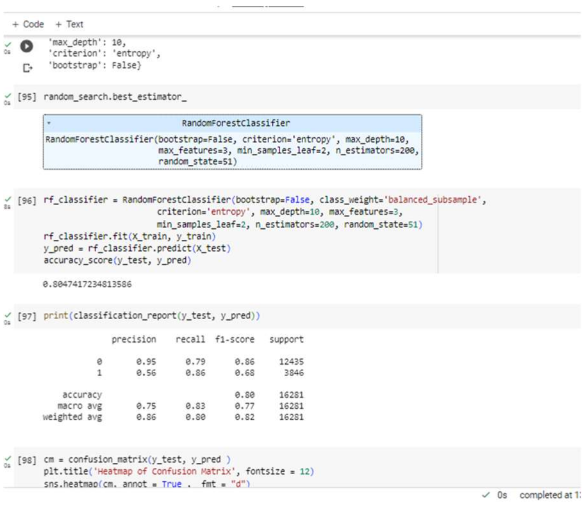
Hyperparameter tuning with random forest

```
[92] params={'max_depth':[3,5,10,None],
             'n_estimators':[10,100,200,300,400,500],
             'max_features':[1,2,3],
              'criterion':['gini','entropy'],
              'bootstrap':[True,False],
              'min_samples_leaf': [1,2,3,4],
             }
```

```
[93] # Randomized Search
     from sklearn.model_selection import RandomizedSearchCV
     random_search = RandomizedSearchCV(rf_classifier, param_distributions=params, scoring= 'roc_auc', n_jobs= -1, verbose= 3 ,random_state = 23)
     random_search.fit(X_train, y_train)

     Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

```
[94] random_search.best_params_

     {'n_estimators': 200,
      'min_samples_leaf': 2,
      'max_features': 3,
      'max_depth': 10,
      'criterion': 'entropy',
      'bootstrap': False}
```

```
[95] random_search.best_estimator_
```



```
      'max_depth': 10,
      'criterion': 'entropy',
      'bootstrap': False}
```

```
[95] random_search.best_estimator_
```

```
RandomForestClassifier(bootstrap=False, criterion='entropy', max_depth=10,
                       max_features=3, min_samples_leaf=2, n_estimators=200,
                       random_state=51)
```

```
[96] rf_classifier = RandomForestClassifier(bootstrap=False, class_weight='balanced_subsample',
                            criterion='entropy', max_depth=10, max_features=3,
                            min_samples_leaf=2, n_estimators=200, random_state=51)
     rf_classifier.fit(X_train, y_train)
     y_pred = rf_classifier.predict(X_test)
     accuracy_score(y_test, y_pred)

     0.8047417234813586
```

```
[97] print(classification_report(y_test, y_pred))

                   precision    recall  f1-score   support

               0       0.95      0.79      0.86     12435
               1       0.56      0.86      0.68      3846

        accuracy                           0.80     16281
       macro avg       0.75      0.83      0.77     16281
    weighted avg       0.86      0.80      0.82     16281
```

```
[98] cm = confusion_matrix(y_test, y_pred )
     plt.title('Heatmap of Confusion Matrix', fontsize = 12)
     sns.heatmap(cm, annot = True ,  fmt = "d")
```