**Census Income Prediction**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION** |  |
|  | * 1. General Introduction |  |
|  | * 1. Project Objectives |  |
|  | **SYSTEM** **PROPOSAL** |  |
|  | * 1. Existing System |  |
|  | * + 1. Disadvantages |  |
|  | * 1. Proposed System |  |
|  | 2.2.1 Advantages |  |
|  | * 1. Literature Survey |  |
|  | **SYSTEM DIAGRAMS** |  |
|  | * 1. Architecture Diagram |  |
|  | * 1. Flow Diagram |  |
|  | * 1. UML Diagrams |  |
|  | **IMPLEMENTATION** |  |
|  | * 1. Modules |  |
|  | * 1. Modules Description |  |
|  | **SYSTEM** **REQUIREMENTS** |  |
|  | * 1. Hardware Requirements |  |
|  | * 1. Software Requirements |  |
|  | * 1. Software Description |  |
|  | * 1. Testing of Products |  |
|  | **CONCLUSION** |  |
| **7.** | **FUTURE** **ENHANCEMENT** |  |
| **8.** | **SAMPLE** **CODING** |  |
| **9.** | **SAMPLE** **SCREENSHOT** |  |
| **10.** | **REFERENCES** |  |

**ABSTRACT**

Although the fairness community has recognized the importance of data, researchers in the area primarily rely on UCI Adult when it comes to tabular data. Derived from a 1994 US Census survey, this dataset has appeared in hundreds of research papers where it served as the basis for the development and comparison of many algorithmic fairness interventions. We reconstruct a superset of the UCI Adult data from available US Census sources and reveal idiosyncrasies of the UCI Adult dataset that limit its external validity. Our primary contribution is a suite of new datasets derived from US Census surveys that extend the existing data ecosystem for research on fair machine learning. We create prediction tasks relating to income, employment, health, transportation, and housing. This system considers how automotive insurance providers incorporate machinery learning in their company, and explores how ML models can apply to insurance big data. We utilize various ML methods, such as gradient boosting algorithm and KNN algorithm, to predict the income census.

**CHAPTER 1**

**INTRODUCTION**

* 1. **General Introduction:**

Datasets are central to the machine learning ecosystem. Besides providing training and testing data for model builders, datasets formulate problems, organize communities, and interface between academia and industry. Influential works relating to the ethics and fairness of machine learning recognize the centrality of datasets, pointing to significant harms associated with data, as well as better data practices. While the discourse about data has prioritized cognitive domains such as vision, speech, or language, numerous consequential applications of predictive modeling and risk assessment involve bureaucratic, organizational, and administrative records best represented as tabular data.

When it comes to tabular data, surprisingly, most research papers on algorithmic fairness continue to involve a fairly limited collection of datasets, chief among them the UCI Adult dataset Derived from the 1994 Current Population Survey conducted by the US Census Bureau, this dataset has made an appearance in more than three hundred research papers related to fairness where it served as the basis for the development and comparison of many algorithmic fairness interventions. Our work begins with a critical examination of the UCI Adult dataset, its origin, impact, and limitations. To guide this investigation we identify the previously undocumented exact source of the UCI Adult dataset, allowing us to reconstruct a superset of the data from available US Census records.

This reconstruction reveals a significant idiosyncrasy of the UCI Adult prediction task that limits its external validity. While some issues with UCI Adult are readily apparent, such as its age, limited documentation, and outdated feature encodings, a significant problem may be less obvious at first glance. Specifically, UCI Adult has a binary target label indicating whether the income of a person is greater or less than fifty thousand US dollars. This income threshold of $50k US dollars corresponds to the 76th quantile of individual income in the United States in 1994, the 88th quantile in the Black population, and the 89th quantile among women.

We show how empirical findings relating to algorithmic fairness are sensitive to the choice of the income threshold, and how UCI Adult exposes a rather extreme threshold. Specifically, the magnitude of violations in different fairness criteria, trade-offs between them, and the effectiveness of algorithmic interventions all vary significantly with the income threshold. In many cases, the $50k threshold understates and misrepresents the broader picture.

* 1. **Objectives:**

The main objective of our project is,

* To classify or to predict or to detect the census income effectively.
* To implement the different classification algorithms for better performances.
* To enhance the overall performance for classification algorithms.

**CHAPTER 2**

**SYSTEM PROPOSAL**

* 1. **EXISTING SYSTEM:**

In existing system, automotive insurance providers incorporate machinery learning in their company, and explores how ML models can apply to insurance big data. We utilize various ML methods, such as logistic regression, XGBoost, random forest, decision trees, naïve Bayes, and K-NN, to predict claim occurrence. Furthermore, we evaluate and compare these models’ performances. The results showed that RF is better than other methods with the accuracy, kappa, and AUC values of 0.8677, 0.7117, and 0.840, respectively. And how to handle an imbalanced dataset to prevent bias to a majority class. Applying ML analytics in insurance is the same as in other industries—to optimize marketing strategies, improve the business, enhance the income, and reduce costs. This paper presented several ML techniques to efficiently analyze insurance claim prediction and compare their performances using various metrics. We proposed a solution using ML models to predict claim occurrence in the next year.

**2.1.1 DISADVANTAGES:**

* It doesn’t efficient for large volume of data’s
* Theoretical limits.
* Training time is high.
* The process is implemented without removing unwanted data.
  1. **PROPOSED SYSTEM:**

In this system, the census income dataset was taken as input. The input data was taken from the dataset repository. Then, we have to implement the data pre-processing step. In this step, we have to handle the missing values for avoid wrong prediction, and to encode the label for input data. Then, we have to split the dataset into test and train. The data is splitting is based on ratio. In train, most of the data’s will be there. In test, smaller portion of the data’s will be there. Training portion is used to evaluate the model and testing portion is used to predicting the model. After that, we can implement the feature selection such as correlation. Then, we have to implement the classification algorithm (i.e.) machine learning. The machine learning algorithms such as KNN and gradient boosting. Finally, the experimental results shows that the performance metrics such as accuracy and comparison results.

**2.2.1 ADVANTAGES:**

* It is efficient for large number of datasets.
* Time consumption is low.
* The process is implemented with removing unwanted data.
* Prediction is accurate.

**2.3 LITERATURE SURVEY:**

**2.3.1 Estimates of Income for Small Places: An Application of James-Stein Procedures to Census Data, 2018**

**Author***:* Robert E. Fay III, Roger A. Herriot

**Methodology:**

An adaptation of the James-Stein estimator is applied to sample estimates of income for small places (i.e., population less than 1,000) from the 1970 Census of Population and Housing. The adaptation incorporates linear regression in the context of unequal variances. Evidence is presented that the resulting estimates have smaller average error than either the sample estimates or an alternate procedure of using county averages. The new estimates for these small places now form the basis for the Census Bureau's updated estimates of per capita income for the General Revenue Sharing Program.

**Advantage:**

* Training time is low.

**Disadvantage:**

* It is not efficient for large number of data’s.

**2.3.2 Data analytics to predict the income and economic hierarchy on Census data, 2021**

**Author**: Sharath R; Krishna Chaitanya S; Nirupam K N; Sowmya B J; K G Srinivasa**Methodology:**

The US Census Bureau conducts the American Community Survey generating a massive dataset with millions of data points. The rich dataset contains detailed information of approximately 3.5 million households in regard to who they are and how they live including ancestry, education, work, transportation, internet use and so on. This enormous data encourages the need to know more about the population and to derive insights. The ever demanding requirement in exposing the subtlety in case of economic issues is the motivation behind to construe meaningful conclusions in income domain. Hence the focus is to concentrate on bringing out unique insights into the financial status of the people living in the country. These conclusions delineated might aid in delivering wiser decisions in regard to economic growth of the country. Using relevant attributes, demographic graphs are plotted aiding the conclusions drawn. Also classifications into various economic classes are done using well known classifiers.

**Advantage*:***

* Less Efficiency

**Disadvantage*:***

* Training time is high.

**2.3.3 Data analytics on census data to predict the income and economic hierarchy, 2021**

**Author:** K.G. Srinivasa, R. Sharath, S. Krishna Chaitanya, K.N. Nirupam and B.J. Sowmya

**Methodology:**

The US Census Bureau conducts the American Community Survey generating a massive dataset with millions of data points. The rich dataset contains detailed information of approximately 3.5 million households about who they are and how they live including ancestry, education, work, transportation, internet use and residency. This enormous data encourages the need to know more about the population and to derive insight. The ever demanding requirement in exposing the subtlety in case of economic issues is the motivation behind to construe meaningful conclusions in income domain. Hence the focus is to concentrate on bringing out unique insights into the financial status of the people living in the country. These conclusions delineated might aid in delivering wiser decisions in regard to economic growth of the country. Using relevant attributes, demographic graphs are plotted, aiding the conclusions drawn. Also classifications into various economic classes are done using well known classifiers.

**Advantage*:***

* Training time is low.
* Better performance.

**Disadvantage**:

* It creates a new instance by appropriately combining existing instances, thus making it possible to avoid the disadvantage of over fitting to a certain degree.

**2.3.4 A Statistical Approach to Adult Census Income Level Prediction, 2021**

**Author:** Navoneel Chakrabarty, Sanket Biswas

**Methodology:**

The prominent inequality of wealth and income is a huge concern especially in the United States. The likelihood of diminishing poverty is one valid reason to reduce the world's surging level of economic inequality. The principle of universal moral equality ensures sustainable development and improve the economic stability of a nation. Governments in different countries have been trying their best to address this problem and provide an optimal solution. This study aims to show the usage of machine learning and data mining techniques in providing a solution to the income equality problem. The UCI Adult Dataset has been used for the purpose. Classification has been done to predict whether a person's yearly income in US falls in the income category of either greater than 50K Dollars or less equal to 50K Dollars category based on a certain set of attributes. The Gradient Boosting Classifier Model was deployed which clocked the highest accuracy of 88.16%, eventually breaking the benchmark accuracy of existing works.

**Advantage:**

* We can note that the Sensitivity for all ML models with the unbalanced data is lower than the Sensitivity for balanced data created by different resampling methods

**Disadvantage**:

* It creates a new instance by appropriately combining existing instances, thus making it possible to avoid the disadvantage of over fitting to a certain degree.

**CHAPTER 3**

**SYSTEM DIAGRAMS**

**3.1 SYSTEM ARCHITECTURE:**

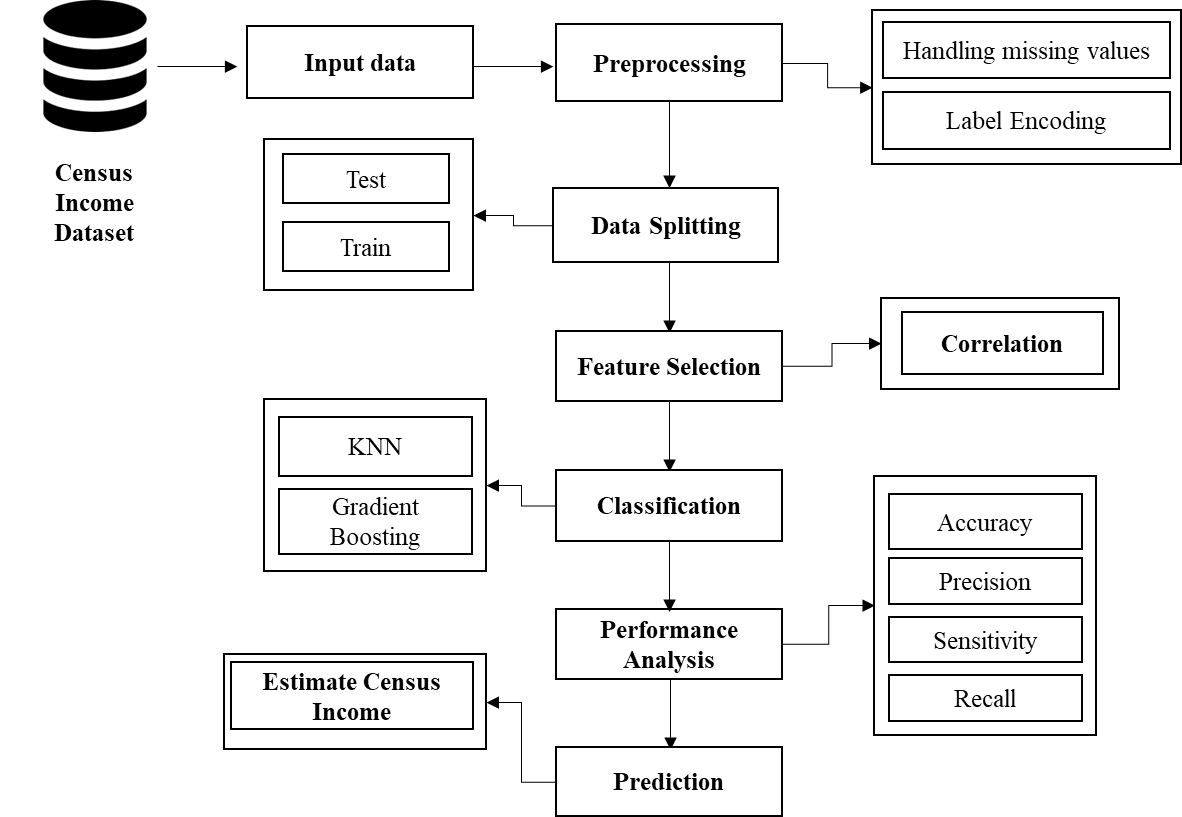
****

FIGURE 3.1: SYSTEM ARCHITECTURE

**3.2 FLOW DIAGRAM**

Input Data

Preprocessing

Data Splitting

Classification

Performance analysis

FIGURE 3.2: FLOW DIAGRAM

**3.3 UML DIAGRAMS:**

**3.3.1 USE CASE DIAGRAM:**

System

User

FIGURE 3.3.1: USE CASE DIAGRAM

**3.3.2 ACTIVITY DIAGRAM:**

Input Data

Preprocessing

Data splitting

Performance metrics

Classification

FIGURE 3.3.2: ACTIVITY DIAGRAM

**3.3.3 SEQUENCE DIAGRAM:**

**Input Data**

**Preprocessing**

**Data splitting**

**Classification**

Select data

Missing value

Test and Train

Load data

Correlation

KNN and GB

FIGURE 3.3.3: SEQUENCE DIAGRAM

**3.3.4 ER DIAGRAM:**

**Data selection**

**Preprocessing**

**Classification**

**Prediction**

FIGURE 3.3.4: ER DIAGRAM

**3.3.5 CLASS DIAGRAM:**

Select data ()

Load data ()

View data ()

**Input Data**

Test ()

**Data Splitting**

Prediction ()

Accuracy ()

Precision ()

Recall()

**Performance**

**Preprocessing**

Missing values ()

Label encode ()

KNN ()

DT ()

**Classification**

Train ()

Correlation ()

FIGURE 3.3.5: CLASS DIAGRAM

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 MODULES:**

* Data selection
* Preprocessing
* Data splitting
* Classification
* Result generation

**4.2 MODULES DESCRIPTION:**

**4.2.1: DATA SELECTION:**

* The input data was collected from dataset repository.
* In our process, census income dataset is used.
* The data selection is the process of predicting the census income.
* The input dataset was taken from dataset repository such as UCI repository.
* In python, with the help of panda’s package, we can read or load our input dataset.
* Our dataset is in the format is ‘.csv’

**4.2.2: DATA PREPROCESSING:**

* Data pre-processing is the process of removing the unwanted data from the dataset.
* Pre-processing data transformation operations are used to transform the dataset into a structure suitable for machine learning.
* This step also includes cleaning the dataset by removing irrelevant or corrupted data that can affect the accuracy of the dataset, which makes it more efficient.
* Missing data removal
* Encoding Categorical data
* Missing data removal: In this process, the null values such as missing values and Nan values are replaced by 0.
* Missing and duplicate values were removed and data was cleaned of any abnormalities.
* Encoding Categorical data: That categorical data is defined as variables with a finite set of label values.
* That most machine learning algorithms require numerical input and output variables.

**4.2.3: DATA SPLITTING:**

* During the machine learning process, data are needed so that learning can take place.
* In addition to the data required for training, test data are needed to evaluate the performance of the algorithm in order to see how well it works.
* In our process, we considered 80% of the input dataset to be the training data and the remaining 20% to be the testing data.
* Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.
* One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
* Separating data into training and testing sets is an important part of evaluating data mining models.
* Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

**4.2.5: CLASSIFICATION:**

* In our process, we have to implement the different classification algorithm such as gradient boosting and KNN.
* **Gradient boosting** is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.
* The **k-nearest neighbors** algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

**4.2.6: RESULT GENERATION:**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

* **Accuracy:**

Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

AC= (TP+TN)/ (TP+TN+FP+FN)

* **Precision**

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

Precision=TP/ (TP+FP)

* **Recall**

Recall is the number of correct results divided by the number of results that should have been returned. In binary classification, recall is called sensitivity. It can be viewed as the probability that a relevant document is retrieved by the query.

Recall=TP/ (TP+FN)

**CHAPTER 5**

**SYSTEM REQUIREMENTS**

**5.1 HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz
* Hard Disk : 200 GB
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 4GB

**5.2 SOFTWARE REQUIREMENTS:**

* O/S : Windows 7.
* Language : Python
* Front End : Anaconda Navigator – Spyder

**5.3 SOFTWARE DESCRIPTION:**

**5.3.1 Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

## **5.3.2 Features of Python**

### **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org) to create games for your computer and for iPhone, iPad, and Android.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

**5.4 TESTING PRODUCTS:**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**5.4.1 UNIT TESTING:**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.

The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

**5.4.2 INTEGRATION TESTING:**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing. ii) Bottom-up integration testing.

**5.4.3 TESTING TECHNIQUES/STRATEGIES:**

* **WHITE BOX TESTING:**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we

Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

* **BLACK BOX TESTING:**

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**5.4.4 SOFTWARE TESTING STRATEGIES**

**VALIDATION TESTING:**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many,

But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer

**USER ACCEPTANCE TESTING:**

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

**OUTPUT TESTING**:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**CHAPTER 6**

**CONCLUSION**

We conclude that, the auto insurance claim dataset was taken as input. The input dataset was mentioned in our research paper. We are implemented the classification algorithms (i.e) machine learning algorithms. Then, machine learning algorithms such as KNN and gradient boosting classification. Finally, the result shows that the accuracy for above mentioned algorithm and estimated the performances metrics such as accuracy for both algorithms and comparison graph.

**CHAPTER 7**

**FUTURE ENHANCEMENT**

Future work may be done in the next directions: Using hybrid classifiers to improve comparison and performance. Furthermore, feature selection approaches may be used to enhance model results and gain a deeper understanding of the important features. It will also be worthwhile to conduct this research for another insurance branch, whether to predict claim occurrences or to predict fraud because these kinds of data always are very heavily unbalanced.

**CHAPTER 8**

**SAMPLE CODING**

#============================= IMPORT LIBRARIES =============================

import pandas as pd

from sklearn import preprocessing

import warnings

warnings.filterwarnings("ignore")

#============================= DATA SELECTION ==============================

dataframe=pd.read\_csv("adult.csv")

print("----------------------------------------------------")

print("Input Data ")

print("----------------------------------------------------")

print()

print(dataframe.head(20))

#============================= PREPROCESSING ==============================

#==== checking missing values ====

print("----------------------------------------------------")

print(" Checking Missing Values ")

print("----------------------------------------------------")

print()

print(dataframe.isnull().sum())

#==== LABEL ENCODING ====

label\_encoder = preprocessing.LabelEncoder()

print("------------------------------------------------------")

print(" Before label encoding")

print("------------------------------------------------------")

print()

print(dataframe['income'].head(10))

print("------------------------------------------------------")

print(" After label encoding")

print("------------------------------------------------------")

print()

dataframe['workclass']=label\_encoder.fit\_transform(dataframe['workclass'])

dataframe['education']=label\_encoder.fit\_transform(dataframe['education'])

dataframe['marital.status']=label\_encoder.fit\_transform(dataframe['marital.status'])

dataframe['occupation']=label\_encoder.fit\_transform(dataframe['occupation'])

dataframe['relationship']=label\_encoder.fit\_transform(dataframe['relationship'])

dataframe['race']=label\_encoder.fit\_transform(dataframe['race'])

dataframe['sex']=label\_encoder.fit\_transform(dataframe['sex'])

dataframe['native.country']=label\_encoder.fit\_transform(dataframe['native.country'])

dataframe['income']=label\_encoder.fit\_transform(dataframe['income'])

print(dataframe['income'].head(10))

#========================= DATA SPLITTING ==============================

print("----------------------------------------------------")

print("Data Splitting ")

print("----------------------------------------------------")

print()

from sklearn.model\_selection import train\_test\_split

X = dataframe.drop('income', axis=1)

y = dataframe['income']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=100)

print("Total no of data's :",dataframe.shape[0])

print()

print("Total no of Train data's :",X\_train.shape[0])

print()

print("Total no of Test data's :",X\_test.shape[0])

# === K NEAREST NEIGHBOUR =====

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

# fit the regressor with x and y data

knn.fit(X\_train, y\_train)

Y\_pred\_knn = knn.predict(X\_test)

from sklearn import metrics

Accuracy\_knn=metrics.accuracy\_score(y\_test,Y\_pred\_knn)\*100

print("----------------------------------------")

print("K NEAREST NEIGHBOUR --> KNN")

print("------------------------------------")

print()

print("1. Accuracy =",Accuracy\_knn,'%' )

print()

print(metrics.classification\_report(y\_test,Y\_pred\_knn))

#========================= CLASSIFICATION ==============================

from sklearn.ensemble import GradientBoostingClassifier

gbt = GradientBoostingClassifier()

gbt.fit(X\_train, y\_train)

y\_pred\_gbt = gbt.predict(X\_test)

from sklearn import metrics

acc\_gbt=metrics.accuracy\_score(y\_pred\_gbt,y\_test)\*100

print("----------------------------------------")

print(" Gradient Boosting Algorithm ")

print("----------------------------------------")

print()

print("1. Accuracy = ", acc\_gbt,'%')

print()

print(metrics.classification\_report(y\_pred\_gbt,y\_test))

print()

#========================= PREDICTION ==============================

print("----------------------------------------------------")

print("Prediction ---> Census Income ")

print("----------------------------------------------------")

print()

print()

for i in range(0,10):

if y\_pred\_gbt[i]==1:

print("-----------------------------")

print()

print([i],"The Census Income = '50K'")

else:

print("------------------------------")

print()

print([i],"The Census Income = '<=50K'")

# === VISUALIZATION ===

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(5, 5))

plt.title("Income")

sns.countplot(x='income',data=dataframe)

plt.show()

import seaborn as sns

plt.figure(figsize=(5, 5))

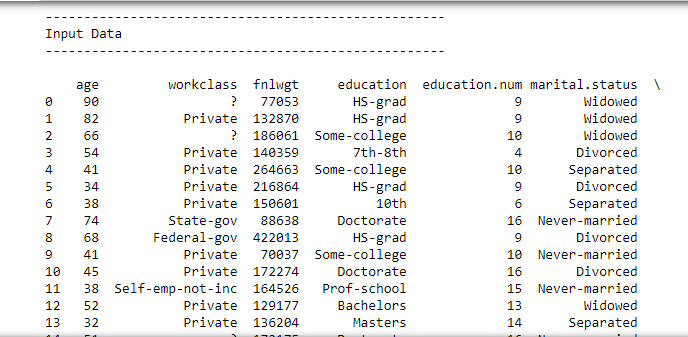
plt.title("Occupation")

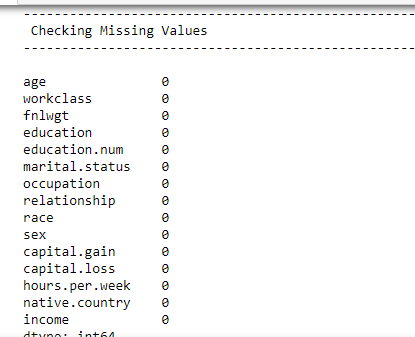
sns.countplot(x='occupation',data=dataframe)

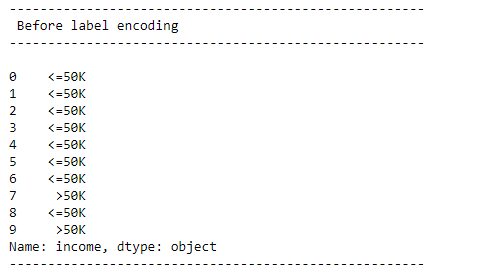
plt.show()

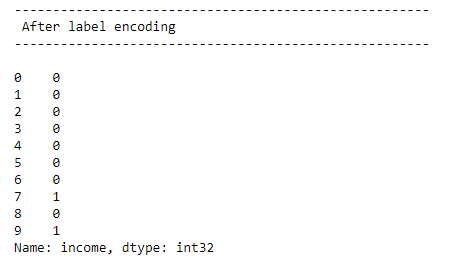
**CHAPTER 9**

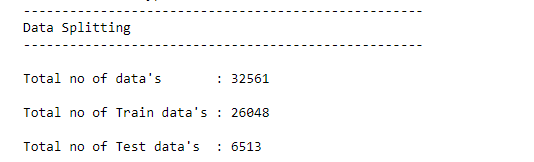
**SAMPLE SCREENSHOTS**

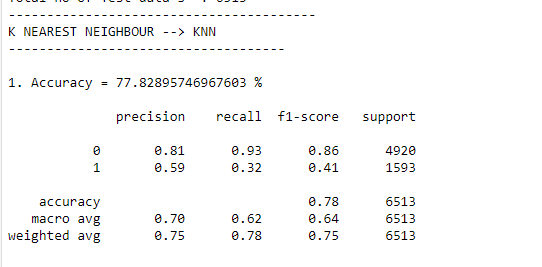


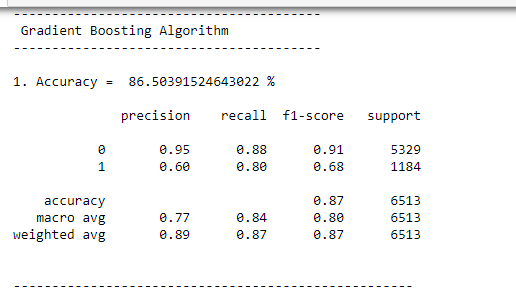


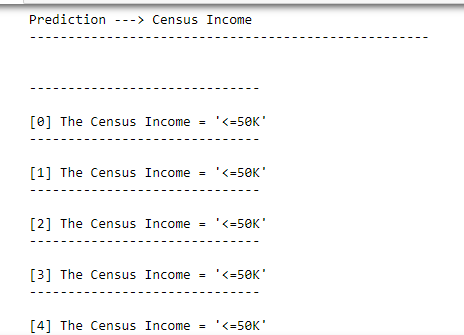


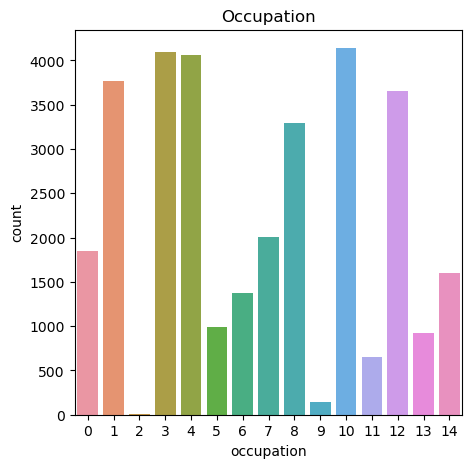












**CHAPTER 10**

**REFERENCES**

[1] R. Abebe, K. Aruleba, A. Birhane, S. Kingsley, G. Obaido, S. L. Remy, and S. Sadagopan. Narratives and counternarratives on data sharing in africa. In Proc. of the ACM Conference on Fairness, Accountability, and Transparency, pages 329–341, 2021.

[2] A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach. A reductions approach to fair classification. In International Conference on Machine Learning, pages 60–69. PMLR, 2018.

[3] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. arXiv preprint arXiv:1907.02893, 2019.

[4] M. Bao, A. Zhou, S. Zottola, B. Brubach, S. Desmarais, A. Horowitz, K. Lum, and S. Venkatasubramanian. It’s compaslicated: The messy relationship between rai datasets and algorithmic fairness benchmarks. arXiv preprint arXiv:2106.05498, 2021.

[5] S. Barocas and A. D. Selbst. Big data’s disparate impact. California Law Review, 104, 2016.

[6] S. Barocas, M. Hardt, and A. Narayanan. Fairness and Machine Learning. fairmlbook.org, 2019. <http://www.fairmlbook.org>.

[7] R. K. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting and mitigating ´ algorithmic bias. IBM Journal of Research and Development, 63(4/5):4–1, 2019.

[8] R. Benjamin. Race after Technology. Polity, 2019.

[9] S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, and K. Walker. Fairlearn: A toolkit for assessing and improving fairness in ai. Microsoft, Tech. Rep. MSR-TR2020-32, 2020.

[10] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in Neural Information Processing Systems, 2016.

[11] J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Fairness, Accountability and Transparency, pages 77–91, 2018.

[12] T. Calders, F. Kamiran, and M. Pechenizkiy. Building classifiers with independency constraints. In In Proc. IEEE ICDMW, pages 13–18, 2009.

[13] A. Caliskan, J. J. Bryson, and A. Narayanan. Semantics derived automatically from language corpora contain human-like biases. Science, 356(6334):183–186, 2017.

[14] A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big data, 5(2):153–163, 2017.

[15] V. Eubanks. Automating inequality: How high-tech tools profile, police, and punish the poor. St. Martin’s Press, 2018.

[16] S. Flood, M. King, R. Rodgers, S. Ruggles, and J. R. Warren. Integrated Public Use Microdata Series, Current Population Survey: Version 8.0 [dataset], 2020. Minneapolis, MN: IPUMS, <https://doi.org/10.18128/D030.V8.0>.

[17] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford. Datasheets for datasets. arXiv:1803.09010, 2018.

[18] M. L. Gray and S. Suri. Ghost work: how to stop Silicon Valley from building a new global underclass. Eamon Dolan Books, 2019.

[19] M. Hardt and B. Recht. Patterns, predictions, and actions: A story about machine learning. https://mlstory.org, 2021.

[20] M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In Proc. 29th NIPS, pages 3315–3323, 2016.