CENSUS INCOME PROJECT

Introduction :

The Census Bureau reports income from several major household surveys and programs. Each differs from the others in some way, such as the length and detail of its questionnaire, the number of households included (sample size), and the methodology used. Census money income is defined as income received on a regular basis (exclusive of certain money receipts such as capital gains) before payments for personal income taxes, social security, union dues, Medicare deductions, etc. Therefore, money income does not reflect the fact that some families receive part of their income in the form of noncash benefits, such as food stamps, health benefits, subsidized housing, and goods produced and consumed on the farm. In addition, money income does not reflect the fact that noncash benefits are also received by some nonfarm residents which may take the form of the use of business transportation and facilities, full or partial payments by business for retirement programs, medical and educational expenses, etc.

Defining the problem:

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)). The prediction task is to determine whether a person makes over $50K a year.
In this blog, we will analyze the Census Dataset from the UCI Machine Learning Repository.We will use Logistic Regression to build the classifier. Its provides the information on the Age,Workclass,Fnlwgt,Education,Education_num,Marital_status,Occupation,Relationship,Race,Sex,Capital_gain, Capital_loss,Hours_per_week,Native_country and Income.we are asked to predict whether Income in US falls in the income category of either greater than 50K Dollars or less equal to 50K Dollars.

Data Pre-processing :

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

- ## Hardware and Software Requirements and Tools Used

- *Tools: Python 3.8.5, Jupyter Notebook, Numpy, Pandas, Matplotlib, Seaborn, Scikit-learn, Scipy*
- *Techniques: Logistic Regression, GaussianNB, SVC, Random Forest Classifier, Ada Boost Classifier, Decision Tree Classifier, Gradient Boosting Classifier.*
- *Hardware: I3 processor, 4GB RAM*

- ## Model/s Development and Evaluation :

  I have used the below models for classification:

- LogisticRegression
- SVC
- RandomForestClassifier
- AdaBoostClassifier
- DecisionTreeClassifier
- *GaussianNB*
- GradientBoostingClassifier
- KNeighborsClassifier

- Identification of possible problem-solving approaches (methods)

- Read the data (from csv)
- Identify the dependent and independent variables.
- Check if the data has missing values or the data is categorical or not.
- If yes, apply basic data preprocessing operations to bring the data in a go to go format.
- Now split the data into the groups of training and testing for the respective purpose.
- After splitting data, fit it to a most suitable model. (How to find a suitable model is answered below)
- Validate the model. If satisfactory, then go with it, else tune the parameters and keep testing. In a few cases, you can also try different algorithms for the same problem to understand the difference between the accuracies.
- From step 7 one can also learn about accuracy paradox.
- Visualize the data.

Data Analysis:

Data analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making.Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains.[2] In today's business world, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively.

Procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.

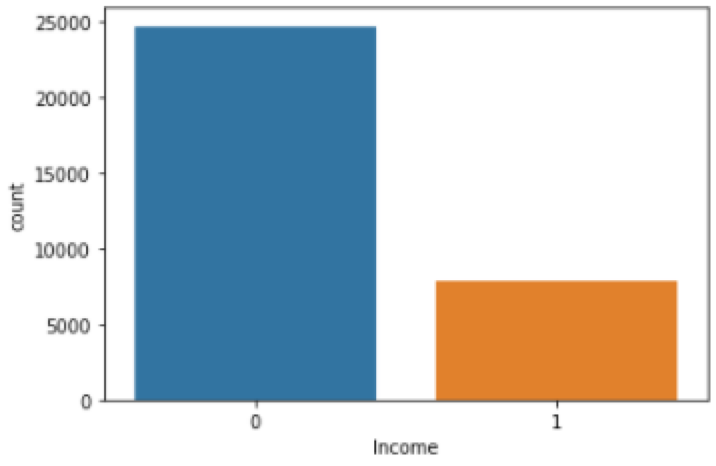
There are several phases that can be distinguished, described below. The phases are iterative, in that feedback from later phases may result in additional work in earlier phases.

EDA :

Once the datasets are cleaned, it can then be analyzed. Analysts may apply a variety of techniques, referred to as exploratory data analysis, to begin understanding the messages contained within the obtained data. The process of data exploration may result in additional data cleaning or additional requests for data; thus, the initialization of the *iterative phases* mentioned in the lead paragraph of this section. Descriptive statistics, such as, the average or median, can be generated to aid in understanding the data. Data visualization is also a technique used, in which the analyst is able to examine the data in a graphical format in order to obtain additional insights, regarding the messages within the data.

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed.

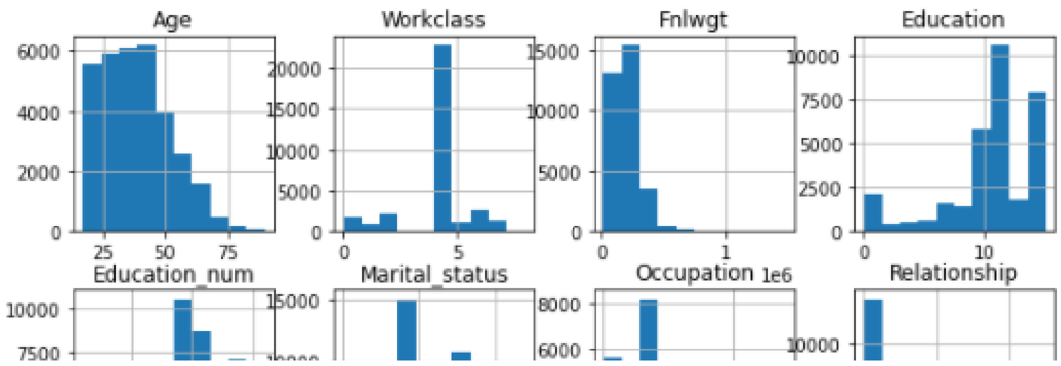
```
sns.countplot(df["Income"]);
```



In census income Target class is imbalance.

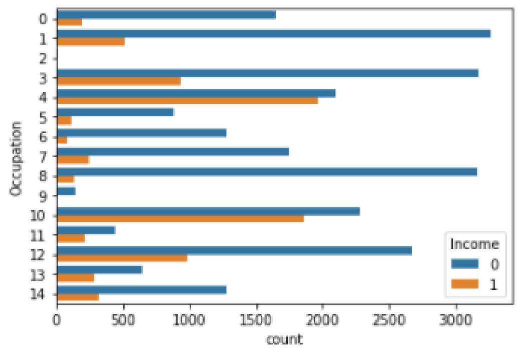
```
df.hist(figsize=(10,10))
```

```
array([[<AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'Workclass'}>,
        <AxesSubplot:title={'center':'Fnlwgt'}>,
        <AxesSubplot:title={'center':'Education'}>],
       [<AxesSubplot:title={'center':'Education_num'}>,
        <AxesSubplot:title={'center':'Marital_status'}>,
        <AxesSubplot:title={'center':'Occupation'}>,
        <AxesSubplot:title={'center':'Relationship'}>],
       [<AxesSubplot:title={'center':'Race'}>,
        <AxesSubplot:title={'center':'Sex'}>,
        <AxesSubplot:title={'center':'Capital_gain'}>,
        <AxesSubplot:title={'center':'Capital_loss'}>],
       [<AxesSubplot:title={'center':'Hours_per_week'}>,
        <AxesSubplot:title={'center':'Native_country'}>,
        <AxesSubplot:title={'center':'Income'}>, <AxesSubplot:>]],
      dtype=object)
```



```
sns.countplot(y='Occupation', hue='Income', data = df)
```

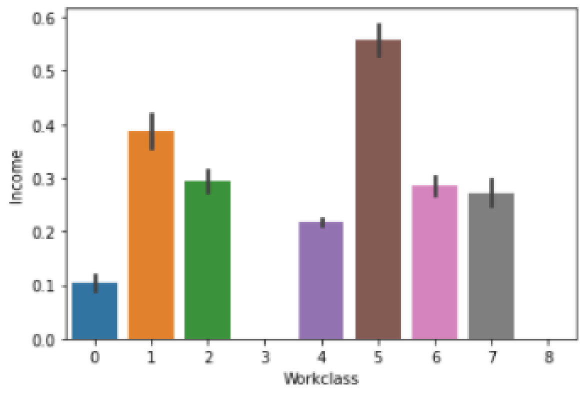
```
<AxesSubplot:xlabel='count', ylabel='Occupation'>
```



In census Income has no missing values.In first plot is showing is workclass of the dataset and second plot is showing Occupation of the dataset.

```
: sns.barplot(x='Workclass', y='Income', data=df)
```

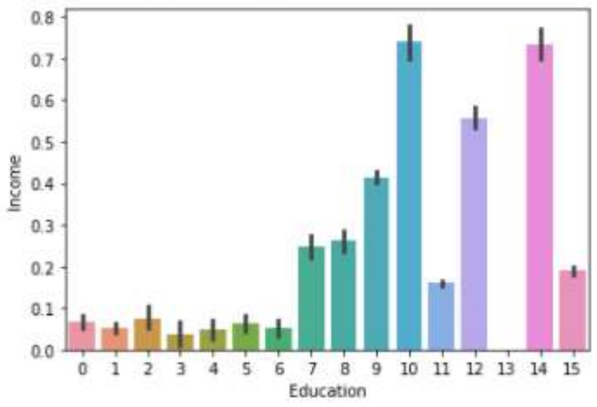
```
: <AxesSubplot:xlabel='Workclass', ylabel='Income'>
```



In census income dataset, in given plot 5 column has most income showing. The lowest income is shown in 0 column.

```
sns.barplot(x='Education', y='Income', data=df)
```

```
<AxesSubplot:xlabel='Education', ylabel='Income'>
```



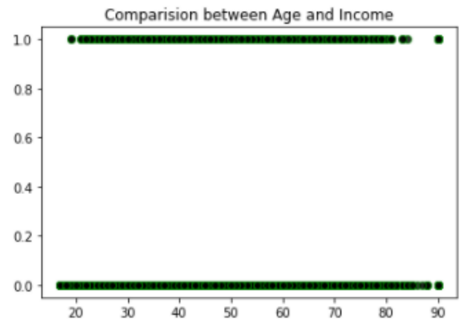
In Census Income dataset, in given plot 14 column has most Income showing.The lowest Income is shown on 3 column.

Bivariate analysis

Comparing multiple variables simultaneously is also another useful way to understand your data. When you have two continuous variables, a scatter plot is usually used. You can use a boxplot to compare one continuous and one categorical variable.
Scatter plots primary uses are to observe and show relationships between two numeric variables. The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole. A scatter plot can also be useful for identifying other patterns in data.

```
plt.scatter(df["Age"],df["Income"],alpha=0.8,c=(0,0,0),edgecolors='g')
plt.title("Comparision between Age and Income")
plt.show()
```

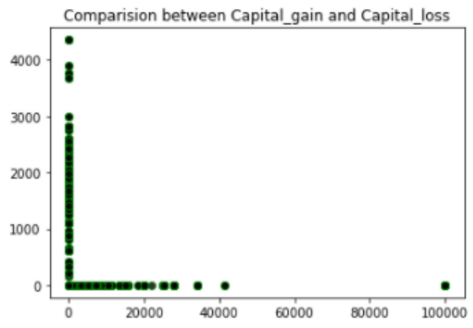
```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```



In these column has outliers are present.

```
plt.scatter(df["Capital_gain"],df["Capital_loss"],alpha=0.8,c=(0,0,0),edgecolors='g')
plt.title("Comparision between Capital_gain and Capital_loss")
plt.show()
```

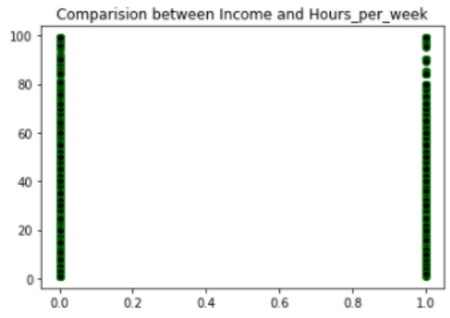
```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```



In this plot has outliers are present.

```
plt.scatter(df["Income"],df["Hours_per_week"],alpha=0.8,c=(0,0,0),edgecolors='g')
plt.title("Comparision between Income and Hours_per_week")
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```



In this plot has outliers are present.

Building Machine Learning Model:

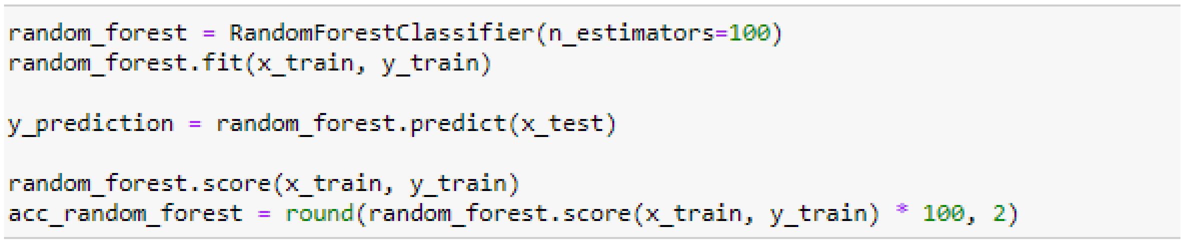
A **machine learning model** is built by **learning** and generalizing from training data, then applying that acquired knowledge to new data it has never seen before to make predictions and fulfill its purpose. Lack of data will prevent you from **building** the **model**, and access to data isn't enough.

The **model building** process involves setting up ways of collecting **data**, understanding and paying attention to what is important in the **data** to answer the questions you are asking, finding a statistical, mathematical or a simulation **model** to gain understanding and make predictions.

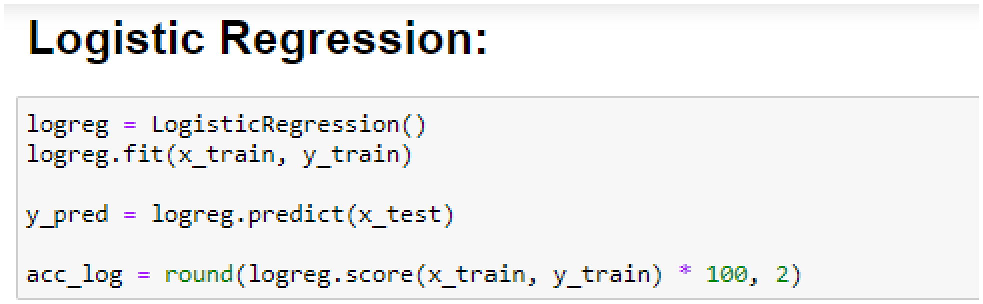
A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

Once you have trained the model, you can use it to reason over data that it hasn't seen before, and make predictions about those data. For example, let's say you want to build an application that can recognize a user's emotions based on their facial expressions. You can train a model by providing it with images of faces that are each tagged with a certain emotion, and then you can use that model in an application that can recognize any user's emotion.

Random Forest:



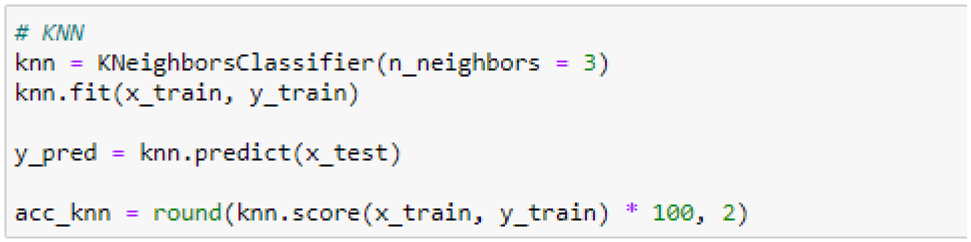
```python
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(x_train, y_train)

y_prediction = random_forest.predict(x_test)

random_forest.score(x_train, y_train)
acc_random_forest = round(random_forest.score(x_train, y_train) * 100, 2)
```



Logistic Regression:

```python
logreg = LogisticRegression()
logreg.fit(x_train, y_train)

y_pred = logreg.predict(x_test)

acc_log = round(logreg.score(x_train, y_train) * 100, 2)
```

K Nearest Neighbor:



```python
# KNN
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)

acc_knn = round(knn.score(x_train, y_train) * 100, 2)
```

Gaussian Naive Bayes: ¶

```
gaussian = GaussianNB()
gaussian.fit(x_train, y_train)

y_pred = gaussian.predict(x_test)

acc_gaussian = round(gaussian.score(x_train, y_train) * 100, 2)
```

Decision Tree

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(x_train, y_train)

y_pred = decision_tree.predict(x_test)

acc_decision_tree = round(decision_tree.score(x_train, y_train) * 100, 2)
```

AdaBoostClassifier

```
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)

y_pred = ad.predict(x_test)

acc_ad = round(ad.score(x_train, y_train) * 100, 2)
```

GradientBoostingClassifier

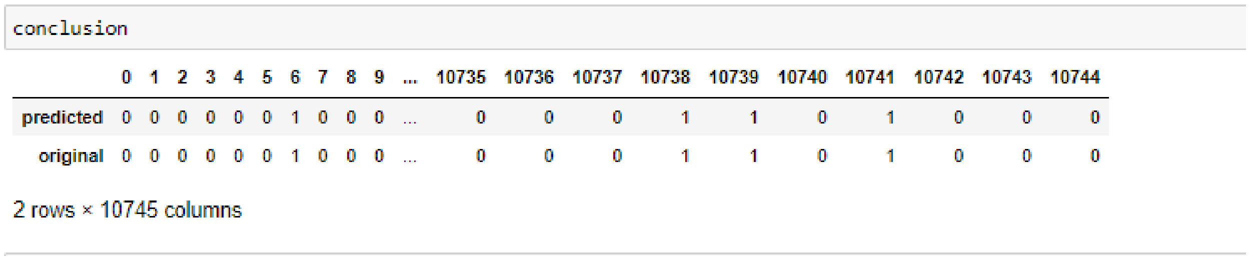
```
gbc=GradientBoostingClassifier()
gbc.fit(x_train,y_train)

y_pred = gbc.predict(x_test)

acc_gbc = round(gbc.score(x_train, y_train) * 100, 2)
```

- **Conclusion:**

Boosting is an ensemble modeling technique which attempts to build a strong classifier from the number of weak classifiers. It is done building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.
Ada Boost was the first really successful boosting algorithm developed for the purpose of binary classification. Ada Boost is short for Adaptive Boosting and is a very popular boosting technique which combines multiple “weak classifiers” into a single “strong classifier”. It was formulated by You Freund and Robert Schapire. They also won the 2003 Gödel Prize for their work.
Ada Boost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances.
Another great quality of ada boost is that they make it very easy to measure the relative importance of each feature. Skearn measure a features importance by looking at how much the tree nodes, that use that feature, reduce impurity on average (across all trees in the forest). It computes this score automatically for each feature after training and scales the results so that the sum of all importance is equal to 1.

Our model predicts 75% of the time, a Income in US falls in the income category of either greater than 50K Dollars or less equal to 50K Dollars (precision). The recall tells us that it predicted the person makes over $50K a year.

The best score in this dataset is Ada boost model. The best score is 86% in this model.



```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	10735	10736	10737	10738	10739	10740	10741	10742	10743	10744
predicted	0	0	0	0	0	0	1	0	0	0	...	0	0	0	1	1	0	1	0	0	0
original	0	0	0	0	0	0	1	0	0	0	...	0	0	0	1	1	0	1	0	0	0

2 rows × 10745 columns