Data Analytics Lab: Final Exam Improvements in the Assignments

Arjav Singh
Metallurgical and Materials Engineering
Indian Institute of Technology Madras
Chennai, India
mm20b007@smail.iitm.ac.in

***Abstract*—This document presents a second attempt on the previous assignments, focusing on improvising them using better techniques and methods.**

I. Introduction

A. Linear Regression

Linear regression is a method employed to uncover the linear connection between a target variable and one or more predictors. It assumes that the model being learned follows a linear pattern based on certain parameters. Three primary types of linear regression exist: Simple and Multiple, where one involves a single independent variable and the other involves multiple independent variables. The third type, Polynomial regression, extends from multiple linear regression by incorporating higher powers of input features. In this context, one variable is the predictor, while the other acts as the response.

Linear regression seeks a statistical relationship rather than a deterministic one between variables. A deterministic relationship implies that one variable can precisely predict or express the other. For instance, using degrees in temperature allows an accurate prediction of Fahrenheit. On the other hand, a statistical relationship, such as that between height and weight, does not precisely determine the connection between two variables.

B. Logistic Regression

Classification is the process of learning a mapping from related input features to discrete data classes or categories. Logistic regression, despite its name, is a technique that can be used to handle binary classification problems. Logistic regression is a process of modelling the probability of a discrete outcome given an input variable. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. The probability is computed by taking the logistic of a linear regression function. The binary logistic model has a dependent variable with two possible values, wherein the corresponding probability of the value labeled '1' can vary between 0 and 1, hence the function that converts the log-odds to probability is the logistic function. The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification, though it can be used to make a classifier, for instance by choosing a cutoff value and classifying the outputs with probability greater than the cutoff as one class, below the cutoff as other.

C. Gaussian Naive Bayes

In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (na¨ıve) independence assumptions between the features (see Bayes classifier). They are among the simplest Bayesian network models,[1] but coupled with kernel density estimation, they can achieve higher accuracy levels.Another assumption made here is that all the predictors have an equal effect on the outcome. Na¨ıve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers. Na¨ıve Bayes models are also known as simple Bayes or independent Bayes. All these names refer to the application of Bayes' theorem in the classifier's decision rule. Na¨ıve Bayes classifier applies the Bayes' theorem in practice. This classifier brings the power of Bayes' theorem to machine learning.

D. Decision Trees

Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks. They are very powerful algorithms, capable of fitting complex datasets. Decision Trees split the instances into two or more homogeneous sets based on most significant splitter / differentiator in input variables. Decision Trees are also the fundamental components of Random Forests, which are among the most powerful Machine Learning algorithms available today. It uses a tree like structure and their possible combinations to solve a particular problem. A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

E. Random Forests

Bootstrap aggregating, also called bagging (from bootstrap aggregating), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach. Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

F. Metrics for model evaluation

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Fig. 1. Confusion Matrix.

1) **Confusion Matrix**: It is used to summarize the performance of a classification algorithm on a set of test data for which the true values are previously known. Sometimes it is also called an error matrix. Terminologies of the Confusion matrix (Figure 1) are:
   - **True Positive**: TP means the model predicted yes, and the actual answer is also yes.
   - **True negative**: TN means the model predicted no, and the actual answer is also no.
   - **False positive**: FP means the model predicted yes, but the actual answer is no.
   - **False negative**: FN means the model predicted no, but the actual answer is yes.

   The rates calculated using the Confusion Matrix are:
   a) **Accuracy**: (TP+TN/Total) tells about overall how classifier Is correct.
   b) **True positive rate**: TP/(actual yes) it says about how much time yes is predicted correctly. It is also called "sensitivity" or "recall."
   c) **False positive rate**: FP/(actual number) says how much time yes is predicted when the actual answer is no.
   d) **True negative rate**: TN/(actual number) says how much time no is predicted correctly, and the actual answer is also no. It is also known as "specificity."
   e) **Misclassification rate**: (FP+FN)/(Total) It is also known as the error rate and tells about how often our model is wrong.
   f) **Precision**: (TP/ (predicted yes)) If it predicts yes, then how often is it correct.
   g) **Prevalence**: (actual yes /total) how often yes condition actually occurs.
   h) **F1-score**: f1 score is defined as the weighted harmonic mean of precision and recall. The best achievable F1 score is 1.0, while the worst is 0.0. The F1 score serves as the harmonic mean of precision and recall. Consequently, the F1-score consistently yields lower values than accuracy measures since it incorporates precision and recall in its computation. When evaluating classifier models, it is advisable to employ the weighted average of the F1 score instead of relying solely on global accuracy.

2) **ROC curve (Receiver Operating Characteristic)**: The Receiver Operating Characteristic (ROC) curve is a useful tool for assessing a model's performance by examining the trade-offs between its True Positive (TP) rate, also known as sensitivity, and its False Negative (FN) rate, which is the complement of specificity. This curve visually represents these two parameters.
   The Area Under the Curve (AUC) metric to summarize the ROC curve concisely. The AUC quantifies the area under the ROC curve. In simpler terms, it measures how well the model can distinguish between positive and negative cases. A higher AUC indicates better classifier performance.
   In essence, AUC categorizes model performance as follows:
   - If AUC = 1, the classifier correctly distinguishes between all the Positive and Negative class points.
   - If 0.5¡ AUC ¡ 1, the classifier will distinguish the positive class value from the negative one because it finds more TP and TN than FP and FN.
   - If AUC = 0.5, the classifier cannot distinguish between positive and negative values.
   - If AUC =0, the classifier predicts all positive as negative and negative as positive.

II. PROBLEM

A. Task in Hand

We have to revisit the assignments and propose better solutions for them with the help of new methods and techniques.

B. Assignment 1: Linear Regression Updates

The assignment revolves around analyzing cancer data aiming to model two continuous variables: average incidence and deaths. Metrics such as Poverty, Income, and Insurance are used as indicators of socioeconomic status. Dependent variables for analysis include Incidence Rate, Average Incidence, Mortality Rate, and Average Deaths across different areas in the United States identified by the FIPS number.

1) Pre-Processing:

We analyze the impact of social status on median income, the sole available social data. A line plot shows varying incomes across communities in different states, indicating higher incomes among Asians and lower incomes among Black communities. If we can demonstrate a substantial influence of median income on Avg Ann Incidence or Deaths, we can consider social status as a valid factor.

Checking for null values, columns related to median income for social groups like Native American, Asian, Black, and Hispanic exhibit substantial missing data (ranging from 20% to 55%), prompting their deletion. Additionally, columns such as Incidence Rate, Avg Ann Incidence, recent trend, mortality rate, and Avg Ann deaths contain non-numeric values. As independent columns lack normalization by population and population data is unavailable, it's preferable to train our model using Avg Ann Incidence and Avg Ann deaths instead of normalized data.

Regarding missing data in Avg Ann Deaths, 325 datapoints contain an asterisk, representing suppressed data due to confidentiality (fewer than 16 cases reported). We plan to address this issue after evaluating other columns. For Avg Ann Incidence, non-numeric data types include '3 or fewer', ' ', which is relatively insignificant compared to the total data.

Feature extraction generates new columns indicating rising and falling recent trends, replacing the recent trend column. Remaining null data requires treatment before model training. Handling methods like removing rows with null values, imputing using state-wise median, or employing a model capable of handling missing data are considered. The latter method is chosen.

2) Visualization:

Scatter plots for Avg Ann Incidence vs. Avg Ann Deaths (Figure 2) show high correlation, suggesting testing on one would yield similar results for the other. Pair plots reveal high correlation among poverty columns and median income. To address multicollinearity, M poverty, F poverty, M with, M without, F with, F without columns are dropped.

3) Statistical Linear Regression Modeling & Updates:

A statistical linear regression model is built with treated parameters as random variables, yielding an adjusted R-squared score of 0.862. Multicollinearity is checked using VIF, identifying a column with a VIF of around 15, necessitating its removal. After dropping this feature, the model achieves greater stability with an adjusted R-squared value of 0.86.

QQ plots comparing the old and new (outlier-free) data reveal a better fit for the t-distribution in the new data, especially at the extremes. The resulting data is outlier-free, making the model more robust. Residual scatter plots show better alignment with the zero line, indicating a better fit of the regression line to the processed data, resulting in lower errors (residuals).

Fig. 2. Assignment 1: Line-plot median income by state

Fig. 3. Assignment 1: Scatter-Plot

C. Assignment 2: Logistic Regression

We are presented with a problem involving the biggest shipwreck of the history, the shipwreck of Titanic. On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

We are given attributes of the passengers of the Titanic including Name, Sex, Age, Passenger class, Number of Siblings/ Spouse onboard, Parents and children onboard, Ticket number,

Fig. 22. Assignment 1: Pair-plot

Fig. 23. Assignment 1: Scatter-plot

Fare, Cabin, and the port where they embarked and the final dependent column being whether the particular passenger survived or not. We need to build a predictive model that answers the question: "what sorts of people are more likely to survive?" using the given passenger data, and discuss any insights and observations.

1) Data Pre-processing & Feature Generation: We start with reading the data to a pandas dataframe. We observe a total of 891 data points, with 12 columns in total, including the different features related to the person onboard. On visualizing the distributions of the people survived we see that around 38% of the total passengers survived the wreck. (Figure 1) We then move on to checking for the null values in the data and find that we have three columns Age, Cabin and Embarked with missing data.

Handling Age, we see that there are a total of 177 missing values that we need to take care of. We tend to check the correlation of Age with the other columns and find out that Pclass has the highest absolute correlation of about 0.37, thus we group age by Pclass and Sex and impute the missing values by the median computed for each group.

Fig. 24. Assignment 1: Scatter-plot

Fig. 25. Assignment 1: Scatter-plot

We then check for the missing values in Embarked and see only two values are missing. Checking Cabin we see that most of the values (77%) are missing. Thus we need to create a new class M (missing), and also extract the cabin class from the cabin column which includes cabin class followed by number. On visualizing the final Cabin column we see that most of the people in the missing class belong to the Pclass 3 and have little chance of survival. We see a higher chance of survival in most of the other cabin classes, with the least being in cabin A and the highest chance being in the cabin class B. We also observe that most of the people in the classes C, E, G, D, A, B belong to the Pclass 1, and have high chances of survival, thus indicating that Pclass might be a factor resulting in higher chances of survival.

On exploring the feature Embarked, we see that it contains three different classes S, C and Q. while most of the people embarked from the S class and majority of them being from Pclass 3. The Passengers from C look to be lucky as a good proportion of them survived. The reason for this maybe the rescue of all the Pclass1 and Pclass2 Passengers. The Embark S looks to the port from where majority of the rich people boarded. Still the chances for survival is low here, that is because many passengers from Pclass3 around 81% didn't survive. Port Q had almost 95% of the passengers were from Pclass3.

We move on to feature generation, we start with checking that many people had the same Ticket numbers, thus we create a new feature representing the number of passengers having the same ticket number, which might be a representation of what was the size of the group in which they were travelling. On careful observation we see that different ticket frequency has different rates of survival with the highest being for the group of two and the chance for groups of more than 4 being very less.

Fig. 26. Assignment 1: Scatter-plot

Fig. 27. Ass 1: Boxplot before vs after

Fig. 28. Ass 1: QQ plot before vs after

Fig. 29. Ass 1: Residual Plot before vs after

Following that we extract the title for each of the passengers from the Name column and find out that people with different titles, had different chances of getting away. We create three categories with the largest being Mr, then clubbing Miss, Mrs and Miss into one, followed by Master and the others grouped into one. We can see that most of the people are form Mr group where the probability is low while the probability of Miss, Mrs, Ms and Master is higher for survival. We have very less people belonging to other classes. We also create a new feature followed from the title classes as Married wherein every passenger with Mrs class is termed as married and we see that the probability of survival for the married class is high.

We created a Child feature for people below 18 years of age, now we can see that children have a higher chance of getting out (Figure 6). We also create a new feature Family size, which is essentially No. of Parents + Children + Siblings + Spouse + 1, post which we create three different categories for the family size class as Alone, Small and Big. We see that people with small family size have the highest chance of Surviving, and the ones with a big family, which the lease chance.

D. Visualization

1) Visualization: We start by creating countplots for the categorical variables with hue as the target columns, to check for the distribution of each of the categories in the classes, plus commenting on whether the different categories have different survival probabilities. Starting with Pclass, We see

Fig. 30. Ass 1: Residual Scatter before vs after

Fig. 31. Ass 2: Embarked

that the people belonging to the Pclass 1 have a higher chance of survival, which keeps on decreasing as we go down the classes. Following which we explore Sex, wherein we observe Proportion of male and female: 2/3 vs 1/3. Male is much less likely to survive, with only 20% chance of survival. For females, ¿70% chance of survival. Obviously, Sex is an important feature to predict survival.

We then explore Age , which is not a categorical variable, thus we create histogram and boxplots for the same, resulting in insights such as passengers are mainly aged 20-40 and younger passengers tends to survive. on exploring SibSp we

Fig. 32. Ass 2: Count-plot for Ticket Frequency

Fig. 33. Ass 2: Boxplot of various titles

see that most of the passengers are travelling alone and the ones with 1 sibling/spouse are more likely to survive. Visualizing Parch we see that ¿70% passengers travel without parents/children, wherein passengers travelling with parents/ children are more likely to survive than those not. Creating distplots and boxplots for Fare, we see that there are people paying too much for their tickets (outliers) and we can see that money gets you a better chance of survival as evident from the graphs.

Finally as part of postprocessing, we convert the age variable into bins of 5 and also convert the Fare variable into bins of 4. Finally we create dummies for the categorical variables so that they are meaningfull to the machine.

2) Statistical Logistic Regression Modelling & Updates: We use the Statsmodels library in python to model the Logistic regression as it gives us the added benefits of getting the significance values of the regression coefficients. Once we have processed all the independent variables we feed them into the statsmodels logit formula, and then proceed to analyze the summary. We get a Log-Likelihood value of -350, with the method being maximum likelihood estimation. We see many

Fig. 34. Ass 2: Countplot - Married or Not

Fig. 35. Ass 2: Countplot - Child or Not

coefficients having p values of greater than 0.05 (corresponding to 95% significance), suggesting that the coefficients are not statistically significant. We see a pseudo R squared value of 0.4089 and an accuracy value of 0.79 at the threshold of 0.41.

We now add a new feature called AgeClass which is the multiplication of Pclass and Age band, hence adding the notion of polynomial features to the setting. As observed we see that this is a class imbalance problem thus we try our hands on upsampling and downsampling using sklearn's resample. Thus from the original 891 samples with 38 % as the positive, we get 1098 samples for the upsampled dataset and 684 for the downsampled dataset as this is done using sampling the already available minority class with or without replacement. Training on the upsampled set we get a Pseudo R squared of 0.45 and 0.448 for the downsampled set. This is quite an improvement over the original value of 0.4089, which is achieved using only simple bootstrapping.

Fig. 36. Ass 2: Countplot for Family Size

Fig. 37. Ass 2: Countplot for Passenger Class

E. Assignment 3: Naive Bayes Classifier

We are presented with a problem involving census data of certain given attributes of the people of various countries including Sex, Age, education, Marital Status, Occupation, Relationship, Finalwgt, Race, Captial gain & loss, working hours per week, workclass and their native country. Using all these features, we are to predict whether the given person makes over 50K per year or not, and also identify relations between the features and the target class. We also see that this is a class imbalanced dataset with only 24 percent of the people with salary more than 50K, but as we have already looked at upsampling and downsampling, we won't be using it here, and will resort to newer techniques.

1) Data Pre-processing: Out of the total 15 features, 9 are categorical and 6 are numerical. We then move on to checking for the null values in the data and find that we do not have any NaN values, but three of the columns Workclass, Occupation and native country have '?' marks as some data points, which need to be treated. The first step is to replace these '?' marks with NaN values and then finally imputing them with the mode

Fig. 38. Ass 2: Countplot - No. of Siblings/Spouse

Fig. 39. Ass 2: Probability of Survival vs No. of Parents/Siblings

of each of the columns as the number of null values is very low and thus no special treatment is necessary. We then check for the cardinality of each of the categorical features, that is the number of different unique values each feature can take as a higher number can cause problems, we find that most of the features do not have more than 7 attributes, wherein the native country feature has the highest number.

2) Visualization and Feature Generation: Now we move on to visualizing each of the features one by one. Starting with workclass, we see that the main types are government employees, Private, Self employed, without pay and never worked classes, with the rate of higher income being different for each of the classes, with the highest rate for self employed inclusive, and the least for private. Moving on to education, the main classes are university level, school level, and postgraduate level, with the trend being lesser income for lesser level and the ones with doctorate and professional school enjoying the highest of salaries.

Moving on to marital status, we see that people married and with spouse enjoy the highest of incomes, while all the

Fig. 40. Ass 2: Boxplot - Fare

Dep. Variable:	Survived	No. Observations:	891
Model:	Logit	Df Residuals:	866
Method:	MLE	Df Model:	24
Date:	Wed, 01 Dec 2021	Pseudo R-squ.:	0.4089
Time:	05:55:35	Log-Likelihood:	-350.70
converged:	True	LL-Null:	-593.33
Covariance Type:	nonrobust	LLR p-value:	1.920e-87

BEFORE

Dep. Variable:	Survived	No. Observations:	1098
Model:	Logit	Df Residuals:	1047
Method:	MLE	Df Model:	50
Date:	Thu, 02 Dec 2021	Pseudo R-squ.:	0.4493
Time:	03:25:23	Log-Likelihood:	-419.10
converged:	False	LL-Null:	-761.08
Covariance Type:	nonrobust	LLR p-value:	3.453e-112

AFTER

Fig. 41. Ass 2: Model stats before vs after

others having very low possibilities of high income. Checking occupation, we see that the number of classes increases tremendously, with each class enjoying different rates, with Executive managers and professional speciality enjoying the highest incomes. Following this, we move on to relationship feature, and find out that husbands and wifes have the highest probability of having high income, as evident in the society, while single people without family or unmarried do not have such high incomes. Moving on to race, we see a bias towards white people having higher salaries, as compared to the others. A yet disturbing fact is seen when sex is visualized, with males enjoying more income on average than females.

Fig. 42. Ass3: Confusion Matrix Before vs After

Fig. 43. Ass3: Count-plot Marital Status

On exploring Age using histograms and boxplots, we see that people of the higher age in general are seen to have higher incomes, which is explainable as most people are not paid well in the starting of their careers. Using this information, we create a new variable isChild that specifies if the person is less than 24 years of age, and we see that children have zero chance of higher income, from the given data. Post this, we also create a feature senior citizen, which is ticked after being 50 years of age or more. We see that senior citizens have more chance of having higher incomes.

We then move forward to education-num which specifies the number of education levels, and see a trend that higher income is captured by people with higher education levels, owing to which we create a new feature highly educated. Similarly, we also explore hourspw which is number of hours worked per week, and see that the people with higher working hours usually tend to have higher incomes, thus creating a new variable work more.

3) Guassian Naive Bayes Classifier Modelling & Updates: We now move on to modelling using the sklearn library's implementation of the gaussian naive bayes classifier. The first step is to divide the data into a train and a validation set, so that we could test on unseen data. The next step is to scale the features for which we use sklearn's robust scaler. The next task is to get a model with the right set of hyperparameters, which is decided by using randomized search croos validation technique from sklearn's model selection toolkit. The hyperparameter being fine-tuned is var smoothing in the logarithmic scale. Moving on to the updates we create bins for

Fig. 44. Ass3: Count-Plot Relationship

Fig. 45. Ass3: Countplot Race

education level and hours worked per week, categorizing them into Low, Medium, High and VeryHigh. We also create a new feature as Occupation Category, with two values as lowskill and highskill, and same with Race category with two values as white or other.

Post this we encode the variables and perform all the preprocessing so as to convert the data into model readable form. Following this we explore a new technique called feature selection using variance threshold as explained in the section 2. This technique helped us to check for the important features and then select them, thus keeping only the most important ones and helping to reduce the variance. Finally we are left with a total of 24 columns after the selection. Post this we train our Naive Bayes model and use random search to find the best hyperparameters and thus get the best performance. We get an updated train and test set accuracy of 0.829 and 0.833 as compared to 0.823 and 0.825 before the updates. We also see improvements in the TP, TN, FP and FNs and get better f1 score of 0.64 as compared to 0.61 initially.

F. Assignment 4 & 5: Tree Methods

We tackle both assignment 4 and 5 in this section as both are based on the same dataset containing various parameters of a car such as buying price, maintenance cost,no. of doors, person capacity, luggage boot size and safety rating with the target being to classify the car as unacceptable, acceptable, good or very good. Thus this a multiclass classification problem with

Fig. 46. Ass3: Countplot - Sex

Fig. 47. Ass3: Countplot Children

Fig. 48. Ass3: Countplot Senior Citizen

Fig. 49. Ass3: Education Number distribution

an imbalanced dataset with target percentages as 3.8, 4, 22.2 and 70 percent.

1) Data Pre-processing: We start with reading the data to a pandas dataframe. We observe a total of 1728 data points, with 7 columns in total, including the different features related to the car. On visualizing the distributions of the target variable, we see a multi-class imbalanced dataset problem, wherein around 70% of the total cars are classified unacceptable, 22.2% just acceptable, 4% good and 3.8% very good (Figure 1). We see that all the 6 features are categorical and most certainly ordinal. We then move on to checking for the null values in the data and find that we do not have any NaN values. We then go on to checking for any features with high cardinality, that is the number of different unique values each feature can take as a higher number can cause problems, but then find out that most of the features have 3/4 unique classes, and also most of the classes are balanced, hence we conclude that this is good clean data.

2) Exploratory Data Analysis: We start with a univariate analysis, wherein we create histplots for each of the six features with hue as the target column, using the seaborn library. We see that some of the classes in some features have certain target classes missing which is essentially very useful for our decision tree model as out model tries to get pure leaves. For e.g. considering buying, we see that high and very high buying costs only leads to unacceptable and acceptable cars; small luggage boot capacity leads to not being very good; 2 person cars are only seen as unacceptable, very high maintenance cars leads to unnaceptable and acceptable, while high maintenance costs lead to no very good classified cars. We see that low safety cars are unacceptable too.

Moving on to bivariate analysis, wherein we create stripplots for each of the 15 feature pairs with the hue as the target column, using the seaborn library, using jitter and dodge values as true. From the analysis we see a similar trend as the univariate analysis wherein some target classes are missing which is expected. I have provided the whole analysis for further observation.

3) Postprocessing: As the dataset has categorical features and hence they needs to be encoded in the appropriate form. There are two main method of encodig:

1. One hot encoding
2. Label encoding

As we have categorial features that are ordinal in nature i.e that can be ranked (ordered) hence label encoding will solve our purpose.Had there been nominal features we could have preferred one hot encoding. We use category encoders library for this.

Following this we split the data using a 70/30 split, with the final dataset being 1209 examples in training set and 519 examples in crossvalidation set.

4) Decision Tree Modelling & Updates: We start with modelling Decision tree classifier first. We use grid search for finding the best parameters four our decision tree model and checking for model classification accuracy on the cross validation dataset. Starting with class weight hyperparameter,

Fig. 50. Ass3: Countplot Highly Educated

Fig. 51. Ass3: Hours per week distribution

Fig. 52. Ass3: Count-Plot Working More

```
accuracy_score on train dataset :  0.8292383292383292
accuracy_score on test dataset :  0.8334527587265841
```

BEFORE

```
accuracy_score on train dataset :  0.8229203229203229
accuracy_score on test dataset :  0.8257754120176067
```

AFTER

Fig. 53. Ass3: Accuracy Before vs After

we keep it as balanced as this automatically calculates the class weights according to their distribution in the train dataset. number of jobs is kept as -1 so that it chooses all the CPU cores available for fitting the model. The scoring metric used by us is accuracy. We use grid search for finding the best parameters by defining a range to check in. The parameters are Max depth, The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min samples split samples. Min samples split: The minimum number of samples required to split an internal node: If int, then consider min samples split as the minimum number. If float, then min samples split is a fraction and ceil(min samples split * n samples) are the minimum number of samples for each split. Min samples leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min samples leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression. Finally we get the best parameters as max depth 9, min samples leaf 2, min samples split 5.

On initial analysis, without the updates we get train accuracy scores of 0.983 and test accuracy scores of 0.969 for the Decision tree model (Assignment 4). As an update step we create polynomial features using the Polynomial Features method of the sklearn library. We set the degree hyperparameter to 4 so that we could get all the feature combinations upto degree 4. Using this we convert the initial 6 features to a total of 207 features. Finally applying the decision tree classifier on the new dataset we get an improved score of 0.987 on the train set and 0.977 on the test set which is about a 1 percent increase which is very significant. We also try SMOTE for upsampling which results into a conversion to a 3408 sample dataset after upsampling and leads to the test set score of 0.971 which is not an improvement over the previous score but an improvement over the initial vanilla model.

5) Random Forest Modelling & Updates: Then we move on to modelling using the Random Forest classifier which is a bagged version of Decision trees. In this case we use randomized search as this does not search the whole space but tries to get to the optimal using random sampled values of hyperparameters as this model is expensive to train. The hyperparameters being searched are: n estimators = number of trees in the forest, max features = max number of features considered for splitting a node, max depth = max number of levels in each decision tree, min samples split = min number of data points placed in a node before the node is split, min samples leaf = min number of data points allowed in a leaf node, bootstrap = method for sampling data points (with or without replacement).Fitting 3 folds for each of 100 candidates, totalling 300 fits we find the best parameters as 'n estimators': 800, 'min samples split': 3, 'min samples leaf': 1, 'max features': 'auto', 'max depth': None, 'bootstrap': False. Using this we are able to achieve a train set accuracy of 1 and test accuracy of 0.969. We then proceed to check for the feature importance as this is provided by the implementation of random forests in the sklearn library. We see that safety has the highest importance with doors having the least.

Moving on using the same treatement on the Random Forest Model and using Random Search for looking for the best hyperparameters, we get a score of 1 on the train set and

BEFORE	precision	recall	f1-score	support
<=50K	0.89	0.89	0.89	7455
>50K	0.64	0.64	0.64	2314
accuracy			0.83	9769
macro avg	0.76	0.76	0.76	9769
weighted avg	0.83	0.83	0.83	9769

AFTER	precision	recall	f1-score	support
<=50K	0.87	0.92	0.89	7455
>50K	0.68	0.55	0.61	2314
accuracy			0.83	9769
macro avg	0.78	0.74	0.75	9769
weighted avg	0.82	0.83	0.83	9769

Fig. 54. Ass3: Classification Report Before vs After

Fig. 55. Ass4&5: Univariate Analysis

Fig. 56. Ass4&5: Multivariate Analysis 1

Fig. 57. Ass4&5: Multivariate Analysis 2

0.985 on the test set from an initial score of 1 on the train set and 0.965 on the test set which is an huge improvement of 2 percent. Here we see that our model is overfitting thus we try remove some features in order to reduce variance. Hence using the feature scores, we eliminate the features with an improtance score of less than 0.01. Using this we are able to achieve a train set accuracy of 0.99 and test set accuracy of 0.96. We then try SMOTE for upsampling and hence are able to get further improvement in the accuracy score to 0.9865.

G. Support Vector Machines

Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter. Classification algorithms in particular are being adopted, which treat the data sets as binary classification problems. Here the legitimate pulsar examples form minority positive class and spurious examples form the majority negative class.

The data set shared here contains 11376 negative examples caused by RFI/noise, and 1,152 real pulsar examples. Each row lists the variables first, and the class label is the final entry. The class labels used are 0 (negative) and 1 (positive). Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile. The remaining four variables are similarly obtained from the DM-SNR curve . These are summarised below:

1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.
9. Class

1) Data Pre-processing: We start with reading the data to a pandas dataframe. We observe a total of 12528 data points, with 9 columns in total, On visualizing the distributions of the

Fig. 58. Ass4&5: Confusion Matrix for Decision Tree

Fig. 59. Ass4&5: Visualization of the decision Tree (look into the codefile for bigger picture)

target variable, we see an imbalanced dataset problem, wherein around 9.2% of the total stars cars are classified as pulsars, 90.8% are not (Figure 1). We see that all the 6 features are numerical. We then move on to checking for the null values in the data and find that three columns have null values in both the train and test datasets, which are Excess kurtosis of the integrated profile, Standard deviation of the DM-SNR curve and Skewness of the DM-SNR curve. Thus, we need to treat these. Hence we compute the median values for each of these columns and impute the missing values in both the training and test sets as these median values, as median values are much less likely to be affected by outliers.

Fig. 60. Ass4&5: Confusion Matrix for Random Forest

Fig. 61. Ass4&5: Feature Importance chart (higher is better)

2) Exploratory Data Analysis: We start with a univariate analysis, wherein we create histplots for each of the eight features with hue as the target column, using the seaborn library. We see that for features as Mean, standard deviation, kurtosis and skewness of integrated profile, the standard deviation is quite high, that is the values are distributed much more in the positive class and are concentrated in the negative class. For the remaining features we see an opposite trend that is the values are distributed much more in the class 0 and are concentrated in the 1 class. We also see that for features like mean, standard deviation of integrated profile and kurtosis, skewness of the DM-SNR curve, the mean is lower in the positive class than the negative class, while it is the opposite for the other features.

Moving on to bivariate analysis, wherein we create pairpolots for each of the columns we see that all the features have a clear boundary of separation and hence we would be able to perform well with our SVM model. We only fee some

```
The model accuracy on train set is 0.9834574028122415

The model accuracy on test set is 0.9691714836223507


Classification Report
              precision    recall  f1-score   support

         acc       0.98      0.90      0.94       118
        good       0.65      0.89      0.76        19
       unacc       1.00      0.99      1.00       358
       vgood       0.86      1.00      0.92        24

    accuracy                           0.97       519
   macro avg       0.87      0.95      0.90       519
weighted avg       0.97      0.97      0.97       519

The model accuracy on train set is 0.9867659222497932

The model accuracy on test set is 0.976878612716763
```

BEFORE

```
Classification Report
              precision    recall  f1-score   support

         acc       0.96      0.93      0.95       118
        good       0.79      1.00      0.88        19
       unacc       0.99      0.99      0.99       358
       vgood       1.00      1.00      1.00        24

    accuracy                           0.98       519
   macro avg       0.94      0.98      0.96       519
weighted avg       0.98      0.98      0.98       519
```

AFTER w/o SMOTE

```
The model accuracy on test set is 0.9710982658959537


Classification Report
              precision    recall  f1-score   support

           0       0.98      0.89      0.93       118
           1       0.66      1.00      0.79        19
           2       0.99      0.99      0.99       358
           3       1.00      1.00      1.00        24

    accuracy                           0.97       519
   macro avg       0.91      0.97      0.93       519
weighted avg       0.98      0.97      0.97       519
```

AFTER w/ SMOTE

Fig. 62. Ass4: Classification Report Before vs After

overlap near the decision boundary regions which can be tuned using regularization values (C values) to find a good fit.

We then move on to creating a correlation map for the dataset and see high correlation between the values, which can cause trouble and should be handled if our accuracy or f1 values are not good enough. Some of the methods can be removing some features or creating hybrid combinations of the features.

3) Postprocessing: We split the data using a 80/20 split, with the final dataset being 10022 examples in training set and 2506 examples in crossvalidation set. We then use the Standard scaler to scale our train and validation values.

4) Support Vector Machine Modelling: We start with modelling using the default hyperparameters usnig the SVC library of sklearn. Default hyperparameter means C=1.0, kernel=rbf and gamma=auto among other parameters.We observe model accuracy score with default hyperparameters: 0.9741 and model f1 score with default hyperparameters: 0.8426.Based on the above analysis we can conclude that our classification model accuracy is very good. Our model is doing a very good job in terms of predicting the class labels. But, this is not true. Here, we have an imbalanced dataset. The problem is that accuracy is an inadequate measure for quantifying predictive performance in the imbalanced dataset problem. So, we must explore alternative metrices that provide better guidance in selecting models. In particular, we would like to know the underlying distribution of values and the type of errors our classifer is making. Thus we check for the f1 score which is helpful with such imbalanced datasets. We see that our model is performing well on the f1 score with a score of 0.84.

But we are not done yeat. We can tune the hyperparameters to get a better score and a better fit. Thus we use Grid Search to search from our defined space of hyperparameters such as trying the different kernels, with C ranging from 0.01 to 10. We also vary the degree for the linear kernel and set the class weightsa s balanced. We also check for the gamma hyperparameter with values as scale and auto. Finally using a 2 fold cross validation we get the best model with 0.87 F1 score on the train and 0.85 on the validation which is an improvement over the initial score of 0.84. The best hyperparameters that are chosen are rbf kernel with auto gamma and C = 2.51 which is higher than the default of 1. Thus we are reducing the regularization.

IV. Conclusion

We have divided this paper into two section, one involving a detailed analysis of the stock time series prediction problem and the other part being improving on the already presented problem statements presented during the whole course of this semester. Starting with the stock analysis problem, involving certain stock attributes for entities like Cognizent, HCL, HDFC, Infosys, SBI and USD. We have been successful in answering the questions like the change in stock price over time, daily return of the stock on average, moving average of the various stocks, correlation between the different stocks and the risk by investing in the different stocks and finally a verdict on investing in what stocks will be a good bet. We conclude on the verdict that Cognizant is a bad deal as the associated risk is high with the expected return being low where infosys can be a better bet with similar risk and much higher expected returns. And SBI has the highest expected return with the highest risk associated, while USD has the lowest associated risk and also the lowest returns.

In the second part, we have applied newer techniques like Outlier Analysis and Treatment using the IQR method, Upsampling and Downsampling using Sklearn's Resample,

Feature Selection using Variance Threshold and Polynomial Feature Transformation and have achieved significant gains in terms of both accuracy metrics and robustness. In this part we have tried to use different techniques for each of the assignment so that we could cover more techniques and not repeat the techniques used for other assignments and thus have kept the analysis crisp and to the point. Having applied these newer techniques and getting the desired gains we are confident on having learnt from the course over the semester.

REFERENCES

[1] https://www.kaggle.com/gunesevitan/titanic-advanced-feature-engineering-tutorial
[2] https://www.kaggle.com/seemamishra33/titanic-survival-prediction?scriptVersionId=37753005
[3] https://www.kaggle.com/lonnieqin/stock-market-analysis-prediction-using-lstm
[4] https://towardsdatascience.com/heres-what-i-ve-learnt-about-sklearn-resample-ab735ae1abc4
[5] https://www.analyticsvidhya.com/blog/2020/11/handling-imbalanced-data-machine-learning-computer-vision-and-nlp/
[6] https://www.kaggle.com/plutosenthil/sklearn-notes#5.PolynomialFeatures
[7] https://www.kaggle.com/srinathsrinivasan1/car-evaluation-using-decision-trees-k-fold
[8] https://www.ritchieng.com/machine-learning-evaluate-classification-model/
[9] https://machinelearningmastery.com/multi-class-imbalanced-classification/
[10] https://www.tableau.com/learn/articles/time-series-analysis
[11] https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation/
[12] https://towardsdatascience.com/why-1-5-in-iqr-method-of-outlier-detection-5d07fdc82097
[13] https://towardsdatascience.com/heres-what-i-ve-learnt-about-sklearn-resample-ab735ae1abc4
[14] https://machinelearningmastery.com/polynomial-features-transforms-for-machine-learning/
[15] https://towardsdatascience.com/how-to-use-variance-thresholding-for-robust-feature-selection-a4503f2b5c3f
[16] Linear regression, Data Analytics Laboratory EE4708, July - November 2021
[17] Linear regression by Prof. Manikandan, Pattern Recognition and Machine Learning, July - November 2021
[18] https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e
[19] https://blog.minitab.com/en/adventures-in-statistics-2/how-to-interpret-regression-analysis-results-p-values-and-coefficients
[20] https://en.wikipedia.org/wiki/Student%27s_t-distribution
[21] https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/
[22] https://stats.stackexchange.com/questions/76441/interpretation-from-loess-graph
[23] https://stats.stackexchange.com/questions/481413/how-to-interpret-this-shape-of-qq-plot-of-standardized-residuals
[24] https://towardsdatascience.com/q-q-plots-explained-5aa8495426c0
[25] https://medium.com/evidentebm/linear-regression-using-statsmodels-d0db5fef16bb
[26] https://data.world/nrippner/cancer-linear-regression-model-tutorial/workspace/file?filename=OLS_regression_walkthrough.ipynb
[27] https://statisticsbyjim.com/regression/heteroscedasticity-regression/
[28] Logistic regression, Data Analytics Laboratory EE4708, July - November 2021
[29] Logistic regression by Prof. Manikandan, Pattern Recognition and Machine Learning, July - November 2021
[30] https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
[31] https://en.wikipedia.org/wiki/Logistic_regression
[32] https://machinelearningmastery.com/confusion-matrix-machine-learning/
[33] Classification, Data Analytics Laboratory EE4708, July - November 2021
[34] Niave Bayes Classifier by Prof. Manikandan, Pattern Recognition and Machine Learning, July - November 2021
[35] https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
[36] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[37] http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/
[38] https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn
[39] https://stackabuse.com/the-naive-bayes-algorithm-in-python-with-scikit-learn/
[40] Random Forests, Data Analytics Laboratory EE4708, July - November 2021
[41] Random Forests by Prof. Manikandan, Pattern Recognition and Machine Learning, July - November 2021
[42] https://en.wikipedia.org/wiki/Gradient_boosting
[43] https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
[44] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[45] https://www.kaggle.com/prashant111/a-guide-on-xgboost-hyperparameters-tuning
[46] https://www.kaggle.com/shubham47/random-forest-classifier-tutorial
[47] https://notebook.community/Aniruddha-Tapas/Applied-Machine-Learning/Classification/Car%20Evaluation%20using%20Decision%20trees%20and%20Random%20Forests
[48] https://www.kaggle.com/mohitcr7/decision-tree-classifier-beginner-level
[49] https://www.kaggle.com/vipulgandhi/a-guide-to-decision-trees-for-beginners
[50] https://en.wikipedia.org/wiki/Decision_tree_pruning
[51] Support Vector Machines, Data Analytics Laboratory EE4708, July - November 2021
[52] Support Vector Machines by Prof. Manikandan, Pattern Recognition and Machine Learning, July - November 2021
[53] https://en.wikipedia.org/wiki/Support-vector_machine
[54] https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
[55] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[56] http://dataaspirant.com/2017/01/13/support-vector-machine-algorithm/
[57] https://www.kaggle.com/prashant111/svm-classifier-tutorial
[58] https://www.ritchieng.com/machine-learning-evaluate-classification-model/
[59] https://en.wikipedia.org/wiki/Kernel_method
[60] https://en.wikipedia.org/wiki/Polynomial_kernel
[61] https://en.wikipedia.org/wiki/Radial_basis_function_kernel

BEFORE

```
The model accuracy on train set is 0.9688850475367329

The model accuracy on test set is 0.9334500875656743

Classification Report
              precision    recall  f1-score   support

         acc       0.89      0.84      0.86       129
        good       0.56      0.90      0.69        20
       unacc       0.98      0.97      0.98       397
       vgood       0.80      0.80      0.80        25

    accuracy                           0.93       571
   macro avg       0.81      0.88      0.83       571
weighted avg       0.94      0.93      0.94       571
```

BEFORE

AFTER w/o SMOTE

```
The model accuracy on train set is 1.0

The model accuracy on test set is 0.9845857418111753

Classification Report
              precision    recall  f1-score   support

         acc       0.97      0.97      0.97       118
        good       0.90      1.00      0.95        19
       unacc       1.00      0.99      1.00       358
       vgood       0.96      0.92      0.94        24

    accuracy                           0.98       519
   macro avg       0.96      0.97      0.96       519
weighted avg       0.98      0.98      0.98       519
```

AFTER w/o SMOTE

AFTER w/ SMOTE

```
The model accuracy on train set is 1.0

The model accuracy on test set is 0.9865125240847784

Classification Report
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       118
           1       0.86      1.00      0.93        19
           2       1.00      0.99      1.00       358
           3       0.96      1.00      0.98        24

    accuracy                           0.99       519
   macro avg       0.95      0.99      0.97       519
weighted avg       0.99      0.99      0.99       519
```

AFTER w/ SMOTE

Fig. 64. Ass5: Classification Report Before vs After

Fig. 63. Ass4: Confusion Matrix Before vs After

Fig. 65. Ass5: Confusion Matrix Before vs After

Fig. 66. Ass6: Univariate Analysis

Fig. 67. Ass6: Univariate Analysis

Fig. 68. Ass6: Multivariate Analysis

Fig. 69. Ass6: Correlation Map

Fig. 70. Ass6: Confusion Matrix

Fig. 71. Ass6: ROC Curve

Assignment 1: Data Analytics Laboratory EE4708

Linear Regression

Madhur Jindal | ME18B059

Reading the data into pandas dataframes

```
# Importing the required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')

import seaborn as sns

#reading the dataset
dataset = pd.read_excel('merged_data.xlsx',index_col=0)

# Visualizing the data
dataset.head()
```

```
  State                              AreaName  All_Poverty  M_Poverty  \
0    AK      Aleutians East Borough, Alaska          553        334
1    AK  Aleutians West Census Area, Alaska          499        273
2    AK      Anchorage Municipality, Alaska        23914      10698
3    AK          Bethel Census Area, Alaska         4364       2199
4    AK         Bristol Bay Borough, Alaska           69         33

   F_Poverty  FIPS  Med_Income  Med_Income_White  Med_Income_Black  \
0        219  2013     61518.0           72639.0           31250.0
1        226  2016     84306.0           97321.0           93750.0
2      13216  2020     78326.0           87235.0           50535.0
3       2165  2050     51012.0           92647.0           73661.0
4         36  2060     79750.0           88000.0               NaN

   Med_Income_Nat_Am  ...  F_Without  All_With  All_Without  fips_x  \
0            54750.0  ...        540      1442         1857    2013
1            48750.0  ...        564      4177         1333    2016
2            53935.0  ...      21393    243173        44638    2020
3            41594.0  ...       1774     13023         4482    2050
4            63333.0  ...         67       768          191    2060

   Incidence_Rate  Avg_Ann_Incidence  recent_trend  fips_y
```

```
Mortality_Rate  \
0               *         3 or fewer             *    2013
*
1               *         3 or fewer             *    2016
*
2            61.5                131        stable    2020
47.3
3            62.7                  6        stable    2050
58.3
4               *         3 or fewer             *    2060
*

  Avg_Ann_Deaths
0              *
1              *
2             96
3              5
4              *

[5 rows x 25 columns]
```

Preprocessing and Data Cleaning

```
dataset.columns
```

```
Index(['State', 'AreaName', 'All_Poverty', 'M_Poverty', 'F_Poverty',
'FIPS',
       'Med_Income', 'Med_Income_White', 'Med_Income_Black',
       'Med_Income_Nat_Am', 'Med_Income_Asian', 'Hispanic', 'M_With',
       'M_Without', 'F_With', 'F_Without', 'All_With', 'All_Without',
'fips_x',
       'Incidence_Rate', 'Avg_Ann_Incidence', 'recent_trend',
'fips_y',
       'Mortality_Rate', 'Avg_Ann_Deaths'],
      dtype='object')
```

```
# Dropping unwanted columns
dataset.drop(['fips_x', 'fips_y'], axis = 1, inplace=True)
```

```
dataset.columns
```

```
Index(['State', 'AreaName', 'All_Poverty', 'M_Poverty', 'F_Poverty',
'FIPS',
       'Med_Income', 'Med_Income_White', 'Med_Income_Black',
       'Med_Income_Nat_Am', 'Med_Income_Asian', 'Hispanic', 'M_With',
       'M_Without', 'F_With', 'F_Without', 'All_With', 'All_Without',
       'Incidence_Rate', 'Avg_Ann_Incidence', 'recent_trend',
'Mortality_Rate',
       'Avg_Ann_Deaths'],
      dtype='object')
```

```
dataset.shape
```

```
(3134, 23)
```

```
#checking the number of null values in each of the features
dataset.isnull().sum()
```

```
State                   0
AreaName                0
All_Poverty             0
M_Poverty               0
F_Poverty               0
FIPS                    0
Med_Income              1
Med_Income_White        2
Med_Income_Black     1210
Med_Income_Nat_Am    1660
Med_Income_Asian     1757
Hispanic              681
M_With                  0
M_Without               0
F_With                  0
F_Without               0
All_With                0
All_Without             0
Incidence_Rate          0
Avg_Ann_Incidence       0
recent_trend            0
Mortality_Rate          0
Avg_Ann_Deaths          0
dtype: int64
```

We see the columns with Med Income Black, Native american, Asian have too many missing values hence we tend to remove these columns.

```
fin_df = dataset.copy()

# Total Mean of the Median income for different communities
fin_df[['Med_Income', 'Med_Income_White', 'Med_Income_Black',
       'Med_Income_Nat_Am', 'Med_Income_Asian', 'Hispanic']].mean()
```

```
Med_Income           46819.837855
Med_Income_White     49490.181992
Med_Income_Black     34750.214137
Med_Income_Nat_Am    43309.998643
Med_Income_Asian     65412.969499
Hispanic             41118.231553
dtype: float64
```

```
# A lineplot of the mean of the median incomes of the different
communities grouped by state
grpb = fin_df.groupby('State')[['Med_Income', 'Med_Income_White',
'Med_Income_Black',
       'Med_Income_Nat_Am', 'Med_Income_Asian', 'Hispanic']].mean()
grpb.plot.line(ylabel = 'Median Income', title = 'Lineplot Median
Income by state for different communities', figsize = (10, 7))
```

```
<AxesSubplot:title={'center':'Lineplot Median Income by state for
different communities'}, xlabel='State', ylabel='Median Income'>
```

We can see that the different social group have different mean incomes for different states, hence if we can prove that the median income is a valid factor determining the Avg Ann Incidence or Deaths, then we can also assume that social status would also be a valid factor !

```
#Dropping the following columns
fin_df.drop(['Med_Income_White', 'Med_Income_Black',
'Med_Income_Nat_Am',
             'Med_Income_Asian', 'Hispanic'], axis=1, inplace=True)

# Checks what all columns in the dataframe contain only numeric values
fin_df.apply(lambda s: pd.to_numeric(s,
errors='coerce').notnull().all())
```

```
State                 False
AreaName              False
All_Poverty            True
M_Poverty              True
F_Poverty              True
FIPS                   True
Med_Income            False
M_With                 True
M_Without              True
F_With                 True
F_Without              True
All_With               True
All_Without            True
Incidence_Rate        False
Avg_Ann_Incidence     False
recent_trend          False
Mortality_Rate        False
Avg_Ann_Deaths        False
dtype: bool
```

```
#this will give us info about the columns that only contain numeric of
null values
fin_df.describe()
```

```
        All_Poverty      M_Poverty      F_Poverty          FIPS  \
count  3.134000e+03    3134.000000    3134.000000   3134.000000
mean   1.522966e+04    6828.800893    8400.855775  30426.019145
std    5.457122e+04   24719.078097   29865.855831  15124.491165
min    1.000000e+01       5.000000       5.000000   1001.000000
25%    1.731250e+03     758.750000     957.000000  19001.500000
50%    4.294000e+03    1925.000000    2372.000000  29180.000000
75%    1.034550e+04    4697.500000    5812.500000  45080.500000
max    1.800265e+06  823612.000000  976653.000000  56045.000000

          Med_Income        M_With      M_Without        F_With  \
count    3133.000000  3.134000e+03    3134.000000  3.134000e+03
mean    46819.837855  4.158963e+04    6930.955329  4.487357e+04
std     12246.380184  1.293894e+05   28686.089548  1.406455e+05
min     19328.000000  3.200000e+01       4.000000  3.300000e+01
25%     38826.000000  4.506750e+03     750.000000  4.657500e+03
50%     45075.000000  1.040450e+04    1763.000000  1.110800e+04
75%     52224.000000  2.788775e+04    4407.250000  2.976475e+04
max    123453.000000  3.904322e+06  997326.000000  4.230137e+06

            F_Without      All_With   All_Without
count    3134.000000  3.134000e+03  3.134000e+03
mean     5968.701021  8.646320e+04  1.289966e+04
std     24657.276997  2.699985e+05  5.331494e+04
min         4.000000  6.700000e+01  8.000000e+00
25%       633.000000  9.173500e+03  1.388250e+03
```

```
50%      1529.000000  2.144800e+04  3.323500e+03
75%      3834.000000  5.756150e+04  8.240000e+03
max    837175.000000  8.134459e+06  1.834501e+06
```

Thus we can see we have to treat the columns [Incidence_Rate, Avg_Ann_Incidence, recent_trend, Mortality_Rate, Avg_Ann_Deaths] for values that are not numeric.

Here as we can see, all the independent columns are not normalized by population and we also do not have population data, it is better to delete the Mortality rate and Incidence rate columns as these are just the average values normalized by population and hence can be dropped !!

```
fin_df.drop(['Mortality_Rate', 'Incidence_Rate'], axis = 1, inplace = True)

# checking for other data than numeric.
[i for i in fin_df['Avg_Ann_Incidence'].unique() if type(i)==str]
```

```
['3 or fewer', '_', '__']
```

On examining, we can see values with '3 or fewer' , _ and '__'

```
[i for i in fin_df['recent_trend'].unique() if type(i)==str]
```

```
['*', 'stable', 'falling', 'rising', '_', '__']
```

```
[i for i in fin_df['Avg_Ann_Deaths'].unique() if type(i)==str]
```

```
['*']
```

```
# Checks for the number of datapoints without numeric data
def f_(x):
    strs = []
    for _, i in enumerate(x):
        try:
            pd.to_numeric(i)
        except:
            strs.append(i)
    print(pd.Series(strs).value_counts())

f_(fin_df['Avg_Ann_Deaths'])
```

```
*    325
dtype: int64
```

We can see 325 datapoints with an asterisk. We can see from the explanation csv that it represents data that has been suppressed due to confidentiality when fewer than 16 cases were reported. So we have two options here, either to delete these row or impute them with the means of each of states they correspond to ! But as this data is not a large fraction of the data we can just delete it !!

```
# which states are associated with the "*"
fin_df.loc[fin_df.Avg_Ann_Deaths=='*', 'State'].value_counts()
```

```
TX    56
NE    37
KS    36
SD    31
ND    30
CO    24
MT    21
ID    15
UT    12
AK    11
GA     8
NM     7
MN     5
OK     5
NV     5
VA     3
OR     3
CA     3
IA     2
WA     2
WY     2
KY     1
MO     1
PA     1
HI     1
MI     1
MS     1
WI     1
Name: State, dtype: int64
```

```
f_(fin_df['Avg_Ann_Incidence'])
```

```
3 or fewer    211
_             192
__             17
dtype: int64
```

Here we convert the '3 or fewer' to 3 and the others to Nan values

```
f_(fin_df['Med_Income'])
```

```
Series([], dtype: int64)
```

```
C:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:9:
DeprecationWarning: The default dtype for empty Series will be
'object' instead of 'float64' in a future version. Specify a dtype
```

```
explicitly to silence this warning.
  if __name__ == '__main__':
```

```
# we create a filter to convert the non-numeric data to either numeric
or to replace by NULL
def filter_(x):
    try:
        return float(str(x).split(' ')[0])
    except ValueError:
        return float('NaN')

# Using the filter on different columns
fin_df['Avg_Ann_Incidence'] = fin_df['Avg_Ann_Incidence'].map(filter_)

fin_df['Avg_Ann_Deaths'] = fin_df['Avg_Ann_Deaths'].map(filter_)
fin_df['Med_Income'] = fin_df['Med_Income'].map(filter_)

print([i for i in fin_df['Avg_Ann_Incidence'].unique() if
type(i)==str])
print([i for i in fin_df['Avg_Ann_Deaths'].unique() if type(i)==str])
#print([i for i in fin_df['recent_trend'].unique() if type(i)==str])
```

```
[]
[]
```

```
# creating columns with Rising and falling !!

def boo(col, chck):
    if col == chck:
        return 1
    return 0

fin_df['Rising'] = fin_df['recent_trend'].apply(lambda x: boo(x,
'rising'))
fin_df['Falling'] = fin_df['recent_trend'].apply(lambda x: boo(x,
'falling'))

fin_df.select_dtypes(include=np.number)
```

```
      All_Poverty  M_Poverty  F_Poverty   FIPS  Med_Income  M_With
M_Without  \
0             553        334        219   2013     61518.0     876
1317
1             499        273        226   2016     84306.0    2470
769
2           23914      10698      13216   2020     78326.0  120747
23245
3            4364       2199       2165   2050     51012.0    6396
2708
4              69         33         36   2060     79750.0     419
124
```

```
...           ...         ...         ...    ...         ...     ...
...
3129         5058       2177       2881  56037     69022.0   19891
3318
3130         1638       1026        612  56039     75325.0    8948
2558
3131         2845       1453       1392  56041     56569.0    9132
1413
3132         1137        489        648  56043     47652.0    3349
691
3133          958        354        604  56045     57738.0    2927
454

      F_With  F_Without  All_With  All_Without  Avg_Ann_Incidence  \
0        566        540      1442         1857                3.0
1       1707        564      4177         1333                3.0
2     122426      21393    243173        44638              131.0
3       6627       1774     13023         4482                6.0
4        349         67       768          191                3.0
...      ...        ...       ...          ...                ...
3129   18600       2683     38491         6001               14.0
3130    9555       1192     18503         3750                5.0
3131    8711       1503     17843         2916                6.0
3132    3490        703      6839         1394                6.0
3133    3087        314      6014          768                4.0

      Avg_Ann_Deaths  Rising  Falling
0                NaN       0        0
1                NaN       0        0
2               96.0       0        0
3                5.0       0        0
4                NaN       0        0
...              ...     ...      ...
3129             9.0       0        0
3130             5.0       0        0
3131             4.0       0        0
3132             5.0       0        0
3133             4.0       0        0

[3134 rows x 15 columns]
```

```
#checking for null values post the pre-processing
fin_df.isnull().sum()
```

```
State                   0
AreaName                0
All_Poverty             0
M_Poverty               0
F_Poverty               0
FIPS                    0
```

```
Med_Income              1
M_With                  0
M_Without               0
F_With                  0
F_Without               0
All_With                0
All_Without             0
Avg_Ann_Incidence     209
recent_trend            0
Avg_Ann_Deaths        325
Rising                  0
Falling                 0
dtype: int64
```

```
fin_df.head()
```

```
  State                              AreaName  All_Poverty  M_Poverty  \
0    AK      Aleutians East Borough, Alaska          553        334
1    AK  Aleutians West Census Area, Alaska          499        273
2    AK      Anchorage Municipality, Alaska        23914      10698
3    AK          Bethel Census Area, Alaska         4364       2199
4    AK         Bristol Bay Borough, Alaska           69         33

   F_Poverty  FIPS  Med_Income  M_With  M_Without  F_With
F_Without  \
0        219  2013     61518.0     876       1317     566        540

1        226  2016     84306.0    2470        769    1707        564

2      13216  2020     78326.0  120747      23245  122426      21393

3       2165  2050     51012.0    6396       2708    6627       1774

4         36  2060     79750.0     419        124     349         67


   All_With  All_Without  Avg_Ann_Incidence recent_trend
Avg_Ann_Deaths  \
0      1442         1857                3.0            *
NaN
1      4177         1333                3.0            *
NaN
2    243173        44638              131.0       stable
96.0
3     13023         4482                6.0       stable
5.0
4       768          191                3.0            *
NaN

   Rising  Falling
```

```
0        0         0
1        0         0
2        0         0
3        0         0
4        0         0
```

```
#Checking for mean values for different features, grouped by state
fin_df.groupby(['State']).mean()
```

```
          All_Poverty     M_Poverty     F_Poverty          FIPS
Med_Income  \
State

AK       2978.782609   1425.260870   1553.521739   2138.217391
66812.565217
AL      13242.686567   5740.791045   7501.895522   1067.000000
37973.134328
AR       7381.920000   3303.280000   4078.640000   5075.000000
36626.480000
AZ      78712.666667  36501.133333  42211.533333   4013.866667
43252.200000
CA     105778.310345  48712.758621  57065.551724   6058.000000
56013.155172
CO      10218.265625   4729.312500   5488.953125   8062.234375
51263.187500
CT      45793.875000  20125.000000  25668.875000   9008.000000
71184.125000
DC     110365.000000  48069.000000  62296.000000  11001.000000
70848.000000
DE      36105.000000  15373.000000  20732.000000  10003.000000
58067.666667
FL      47464.313433  21485.835821  25978.477612  12067.910448
44046.477612
GA      11251.238994   4968.270440   6282.968553  13161.490566
40704.911950
HI      30788.800000  14142.200000  16646.600000  15005.000000
64879.000000
IA       3776.595960   1693.616162   2082.979798  19099.000000
50483.121212
ID       5572.204545   2591.636364   2980.568182  16044.000000
43607.750000
IL      17658.019608   7873.049020   9784.970588  17102.000000
50163.441176
IN      10630.902174   4719.782609   5911.119565  18092.000000
48745.402174
KS       3631.933333   1651.476190   1980.457143  20105.000000
47322.209524
KY       6715.341667   3007.733333   3707.608333  21120.000000
39137.300000
LA      13879.375000   5932.671875   7946.703125  22064.000000
```

```
41411.781250
MA      53493.214286  23007.857143  30485.357143  25014.000000
65974.428571
MD      24033.541667  10270.708333  13762.833333  24044.958333
69200.375000
ME      11267.375000   5047.500000   6219.875000  23016.000000
46141.750000
MI      19480.361446   8863.554217  10616.807229  26083.000000
44464.987952
MN       6858.183908   3145.494253   3712.689655  27087.000000
53926.988506
MO       7964.973913   3553.165217   4411.808696  29117.713043
41755.400000
MS       7945.670732   3430.256098   4515.414634  28082.000000
34938.926829
MT       2689.035714   1246.785714   1442.250000  30056.000000
44497.017857
NC      16674.650000   7399.320000   9275.330000  37100.000000
41784.200000
ND       1504.867925    669.962264    834.905660  38053.000000
55574.867925
NE       2485.107527   1088.548387   1396.559140  31093.000000
48646.129032
NH      11384.000000   5107.000000   6277.000000  33010.000000
60648.900000
NJ      44992.714286  19710.857143  25281.857143  34021.000000
73014.095238
NM      13010.939394   6034.939394   6976.000000  35030.151515
40183.666667
NV      25078.647059  11705.352941  13373.294118  32045.529412
53689.705882
NY      48482.951613  21388.532258  27094.419355  36062.000000
55275.693548
OH      20179.954545   8994.931818  11185.022727  39088.000000
48446.409091
OK       8104.454545   3612.857143   4491.597403  40077.000000
44097.376623
OR      17692.972222   8237.222222   9455.750000  41036.000000
45171.222222
PA      24874.164179  11000.119403  13874.044776  42067.000000
50316.253731
RI      28844.600000  12603.400000  16241.200000  44005.000000
65783.400000
SC      18063.065217   7840.521739  10222.543478  45046.000000
39756.695652
SD       1652.692308    756.615385    896.076923  46067.430769
47679.738462
TN      11764.147368   5240.600000   6523.547368  47095.000000
40168.031579
```

```
TX      17608.074803   7883.543307   9724.531496  48254.000000
46745.778656
UT      12124.172414   5646.620690   6477.551724  49029.000000
54687.034483
VA       6927.651515   3028.363636   3899.287879  51265.848485
53059.212121
VT       4945.214286   2213.642857   2731.571429  50014.000000
52653.500000
WA      23295.179487  10715.025641  12580.153846  53039.000000
50217.076923
WI      10060.388889   4519.638889   5540.750000  55071.097222
50649.000000
WV       5879.709091   2635.436364   3244.272727  54055.000000
39411.818182
WY       2825.869565   1255.739130   1570.130435  56023.000000
57042.304348

               M_With     M_Without         F_With     F_Without  \
State
AK      12366.565217   2917.739130   12147.608696   2411.217391
AL      29389.671642   4678.328358   32530.134328   4294.194030
AR      16012.226667   2881.893333   17237.906667   2603.880000
AZ     178389.533333  35436.066667  191685.466667  30056.200000
CA     269831.241379  52388.275862  287728.534483  43712.500000
CO      34745.546875   5534.781250   36322.625000   4447.265625
CT     194827.125000  19867.500000  212544.125000  15086.125000
DC     276285.000000  22198.000000  323314.000000  14813.000000
DE     132112.666667  13828.000000  146781.000000  11207.666667
FL     112310.552239  27464.761194  124417.328358  24393.179104
GA      24288.446541   5449.440252   26859.364780   5116.691824
HI     123945.400000   8997.200000  130480.400000   7047.000000
IA      14020.090909   1243.060606   14529.464646   1006.444444
ID      15305.886364   2773.363636   15626.613636   2562.863636
IL      53124.647059   7664.294118   57616.421569   6047.813725
IN      29740.010870   4733.456522   31571.097826   4270.260870
KS      11722.238095   1594.009524   12237.314286   1435.695238
KY      15362.841667   2174.991667   16486.958333   1910.525000
LA      28475.515625   5594.515625   31249.015625   5339.109375
MA     218643.642857  10231.071429  237913.857143   6623.428571
MD     104805.500000  12109.708333  116597.083333   9684.958333
ME      35496.687500   4614.687500   38427.937500   3637.187500
MI      51325.879518   6328.108434   55251.277108   5036.240964
MN      28082.068966   2476.908046   29202.885057   1857.229885
MO      21828.939130   3295.878261   23439.895652   3021.800000
MS      14163.085366   2915.890244   15824.743902   2720.536585
MT       7459.232143   1468.642857    7611.232143   1303.982143
NC      39094.400000   7360.030000   43500.040000   6541.590000
ND       6080.886792    679.924528    6035.056604    530.679245
NE       8726.322581   1086.344086    9035.322581    947.000000
```

```
NH      57839.500000   6636.700000   60861.000000   5568.500000
NJ     176838.857143  26562.190476  193311.333333  22173.523810
NM      25007.909091   5452.000000   26911.939394   4740.333333
NV      65350.588235  15792.000000   67480.411765  13947.588235
NY     134038.258065  17297.806452  149030.435484  12977.209677
OH      56197.590909   6936.954545   60783.000000   5660.829545
OK      19749.077922   4244.116883   21039.896104   3907.480519
OR      46073.055556   7253.416667   48939.194444   6089.083333
PA      82057.477612   9146.940299   89112.880597   7370.582090
RI      89317.200000  10941.000000   99037.000000   8290.800000
SC      41092.608696   7752.934783   45848.934783   7033.695652
SD       5532.461538    709.692308    5630.861538    609.646154
TN      27868.221053   4724.884211   30850.915789   3897.357895
TX      39586.787402  10816.748031   41903.157480  10301.303150
UT      42691.068966   6939.689655   43393.172414   6177.206897
VA      25542.757576   3658.446970   27926.901515   3227.848485
VT      20227.000000   1575.571429   21508.642857   1007.142857
WA      76236.564103  11122.410256   79952.794872   9069.102564
WI      35358.930556   3563.430556   37140.777778   2664.125000
WV      14296.309091   1975.145455   15042.254545   1820.836364
WY      10802.956522   1764.608696   10670.782609   1552.869565

            All_With   All_Without  Avg_Ann_Incidence  Avg_Ann_Deaths
\
State

AK      24514.173913   5328.956522          15.130435       19.833333

AL      61919.805970   8972.522388          59.597015       47.552239

AR      33250.133333   5485.773333          35.733333       28.546667

AZ     370075.000000  65492.266667         252.600000      182.666667

CA     557559.775862  96100.775862         294.396552      231.818182

CO      71068.171875   9982.046875          35.281250       39.075000

CT     407371.250000  34953.625000         332.875000      216.875000

DC     599599.000000  37011.000000         351.000000      240.000000

DE     278893.666667  25035.666667         257.666667      188.333333

FL     236727.880597  51857.940299         245.462687      178.029851

GA      51147.811321  10566.132075          39.823899       29.841060

HI     254425.800000  16044.200000         155.800000      134.250000

IA      28549.555556   2249.505051          24.131313       18.051546
```

ID	30932.500000	5336.227273	19.500000	20.586207
IL	110741.068627	13712.107843	91.254902	64.990196
IN	61311.108696	9003.717391	57.771739	43.586957
KS	23959.552381	3029.704762	NaN	20.826087
KY	31849.800000	4085.516667	40.083333	29.008403
LA	59724.531250	10933.625000	54.453125	42.500000
MA	456557.500000	16854.500000	358.785714	247.500000
MD	221402.583333	21794.666667	152.458333	114.083333
ME	73924.625000	8251.875000	82.750000	59.750000
MI	106577.156627	11364.349398	95.012048	71.731707
MN	57284.954023	4334.137931	NaN	28.719512
MO	45268.834783	6317.678261	46.243478	34.333333
MS	29987.829268	5636.426829	30.487805	23.975309
MT	15070.464286	2772.625000	13.321429	13.485714
NC	82594.440000	13901.620000	75.580000	54.840000
ND	12115.943396	1210.603774	9.094340	12.000000
NE	17761.645161	2033.344086	13.473118	15.035714
NH	118700.500000	12205.200000	105.600000	73.600000
NJ	370150.190476	48735.714286	280.904762	195.095238
NM	51919.848485	10192.333333	29.424242	27.653846
NV	132831.000000	29739.588235	NaN	108.500000
NY	283068.693548	30275.016129	219.532258	146.677419
OH	116980.590909	12597.784091	109.670455	84.204545
OK	40788.974026	8151.597403	39.272727	33.791667
OR	95012.250000	13342.500000	74.694444	62.363636
PA	171170.358209	16517.522388	159.074627	116.621212

```
RI     188354.200000  19231.800000         172.800000      124.600000

SC      86941.543478  14786.630435          81.413043       60.826087

SD      11163.323077   1319.338462           8.969231       10.882353

TN      58719.136842   8622.242105          59.515789       45.873684

TX      81489.944882  21118.051181          51.480315       48.020202

UT      86084.241379  13116.896552          22.241379       24.764706

VA      53469.659091   6886.295455          39.659091       30.387597

VT      41735.642857   2582.714286          37.357143       26.500000

WA     156189.358974  20191.512821         110.282051       84.837838

WI      72499.708333   6227.555556          55.888889       41.732394

WV      29338.563636   3795.981818          36.436364       27.436364

WY      21473.739130   3317.478261          12.739130       10.857143


          Rising   Falling
State
AK     0.000000  0.000000
AL     0.000000  0.074627
AR     0.000000  0.040000
AZ     0.066667  0.133333
CA     0.000000  0.258621
CO     0.015625  0.062500
CT     0.000000  0.375000
DC     0.000000  0.000000
DE     0.000000  0.000000
FL     0.000000  0.134328
GA     0.000000  0.050314
HI     0.000000  0.000000
IA     0.060606  0.030303
ID     0.000000  0.000000
IL     0.009804  0.127451
IN     0.032609  0.032609
KS     0.000000  0.000000
KY     0.016667  0.041667
LA     0.000000  0.031250
MA     0.000000  0.142857
MD     0.041667  0.125000
ME     0.000000  0.000000
MI     0.000000  0.072289
```

```
MN     0.000000  0.000000
MO     0.052174  0.034783
MS     0.012195  0.048780
MT     0.035714  0.017857
NC     0.010000  0.040000
ND     0.056604  0.037736
NE     0.010753  0.000000
NH     0.000000  0.100000
NJ     0.000000  0.666667
NM     0.000000  0.121212
NV     0.000000  0.000000
NY     0.032258  0.096774
OH     0.000000  0.113636
OK     0.000000  0.038961
OR     0.000000  0.055556
PA     0.029851  0.029851
RI     0.000000  0.000000
SC     0.043478  0.086957
SD     0.000000  0.030769
TN     0.010526  0.063158
TX     0.007874  0.031496
UT     0.000000  0.068966
VA     0.015152  0.106061
VT     0.000000  0.142857
WA     0.000000  0.282051
WI     0.027778  0.055556
WV     0.018182  0.018182
WY     0.000000  0.043478
```

```
# Creating a correlation matrix to check for the correlation between
different columns
fin_df.corr()
```

```
                    All_Poverty  M_Poverty  F_Poverty      FIPS
Med_Income  \
All_Poverty           1.000000   0.999696   0.999792 -0.059265
0.121617
M_Poverty             0.999696   1.000000   0.998986 -0.060504
0.120250
F_Poverty             0.999792   0.998986   1.000000 -0.058212
0.122692
FIPS                 -0.059265  -0.060504  -0.058212  1.000000
0.069289
Med_Income            0.121617   0.120250   0.122692  0.069289
1.000000
M_With                0.957627   0.957438   0.957343 -0.057703
0.261679
M_Without             0.970175   0.971202   0.968879 -0.051458
0.141092
F_With                0.961131   0.960465   0.961240 -0.057880
```

```
0.255890
F_Without             0.959671   0.960657   0.958413 -0.048607
0.135189
All_With              0.959581   0.959144   0.959502 -0.057803
0.258699
All_Without           0.965834   0.966843   0.964556 -0.050167
0.138437
Avg_Ann_Incidence     0.901260   0.898099   0.903467 -0.071399
0.241756
Avg_Ann_Deaths        0.911911   0.909497   0.913488 -0.060881
0.235420
Rising               -0.019629  -0.019341  -0.019858  0.010595   -
0.010574
Falling               0.269978   0.269447   0.270294 -0.006330
0.145025

                      M_With  M_Without    F_With  F_Without  All_With
\
All_Poverty         0.957627   0.970175  0.961131   0.959671  0.959581

M_Poverty           0.957438   0.971202  0.960465   0.960657  0.959144

F_Poverty           0.957343   0.968879  0.961240   0.958413  0.959502

FIPS               -0.057703  -0.051458 -0.057880  -0.048607 -0.057803

Med_Income          0.261679   0.141092  0.255890   0.135189  0.258699

M_With              1.000000   0.942191  0.999459   0.928710  0.999853

M_Without           0.942191   1.000000  0.941427   0.997856  0.941920

F_With              0.999459   0.941427  1.000000   0.927356  0.999876

F_Without           0.928710   0.997856  0.927356   1.000000  0.928130

All_With            0.999853   0.941920  0.999876   0.928130  1.000000

All_Without         0.936459   0.999542  0.935421   0.999380  0.936045

Avg_Ann_Incidence  0.949095   0.858194  0.953351   0.842368  0.951440

Avg_Ann_Deaths     0.953423   0.871616  0.957293   0.856068  0.955570

Rising             -0.021823  -0.018979 -0.022058  -0.019055 -0.021948

Falling             0.307680   0.261397  0.307576   0.259156  0.307668


                    All_Without  Avg_Ann_Incidence  Avg_Ann_Deaths
Rising  \
```

```
All_Poverty           0.965834           0.901260        0.911911 -
0.019629
M_Poverty             0.966843           0.898099        0.909497 -
0.019341
F_Poverty             0.964556           0.903467        0.913488 -
0.019858
FIPS                 -0.050167          -0.071399       -0.060881
0.010595
Med_Income            0.138437           0.241756        0.235420 -
0.010574
M_With                0.936459           0.949095        0.953423 -
0.021823
M_Without             0.999542           0.858194        0.871616 -
0.018979
F_With                0.935421           0.953351        0.957293 -
0.022058
F_Without             0.999380           0.842368        0.856068 -
0.019055
All_With              0.936045           0.951440        0.955570 -
0.021948
All_Without           1.000000           0.851338        0.864893 -
0.019024
Avg_Ann_Incidence     0.851338           1.000000        0.997726 -
0.027694
Avg_Ann_Deaths        0.864893           0.997726        1.000000 -
0.028952
Rising               -0.019024          -0.027694       -0.028952
1.000000
Falling               0.260500           0.314654        0.306145 -
0.030629

                      Falling
All_Poverty          0.269978
M_Poverty            0.269447
F_Poverty            0.270294
FIPS                -0.006330
Med_Income           0.145025
M_With               0.307680
M_Without            0.261397
F_With               0.307576
F_Without            0.259156
All_With             0.307668
All_Without          0.260500
Avg_Ann_Incidence    0.314654
Avg_Ann_Deaths       0.306145
Rising              -0.030629
Falling              1.000000
```

```
# Using the describe method for the dataframe
fin_df.describe()
```

```
        All_Poverty      M_Poverty      F_Poverty          FIPS  \
count  3.134000e+03    3134.000000    3134.000000   3134.000000
mean   1.522966e+04    6828.800893    8400.855775  30426.019145
std    5.457122e+04   24719.078097   29865.855831  15124.491165
min    1.000000e+01       5.000000       5.000000   1001.000000
25%    1.731250e+03     758.750000     957.000000  19001.500000
50%    4.294000e+03    1925.000000    2372.000000  29180.000000
75%    1.034550e+04    4697.500000    5812.500000  45080.500000
max    1.800265e+06  823612.000000  976653.000000  56045.000000

          Med_Income        M_With      M_Without        F_With  \
count    3133.000000  3.134000e+03    3134.000000  3.134000e+03
mean    46819.837855  4.158963e+04    6930.955329  4.487357e+04
std     12246.380184  1.293894e+05   28686.089548  1.406455e+05
min     19328.000000  3.200000e+01       4.000000  3.300000e+01
25%     38826.000000  4.506750e+03     750.000000  4.657500e+03
50%     45075.000000  1.040450e+04    1763.000000  1.110800e+04
75%     52224.000000  2.788775e+04    4407.250000  2.976475e+04
max    123453.000000  3.904322e+06  997326.000000  4.230137e+06

           F_Without      All_With   All_Without  Avg_Ann_Incidence  \
count    3134.000000  3.134000e+03  3.134000e+03        2925.000000
mean     5968.701021  8.646320e+04  1.289966e+04          71.079316
std     24657.276997  2.699985e+05  5.331494e+04         172.803924
min         4.000000  6.700000e+01  8.000000e+00           3.000000
25%       633.000000  9.173500e+03  1.388250e+03          11.000000
50%      1529.000000  2.144800e+04  3.323500e+03          25.000000
75%      3834.000000  5.756150e+04  8.240000e+03          60.000000
max    837175.000000  8.134459e+06  1.834501e+06        3701.000000

       Avg_Ann_Deaths       Rising      Falling
count     2809.000000  3134.000000  3134.000000
mean        55.822357     0.013720     0.063178
std        127.709719     0.116347     0.243322
min          3.000000     0.000000     0.000000
25%         10.000000     0.000000     0.000000
50%         21.000000     0.000000     0.000000
75%         48.000000     0.000000     0.000000
max       2876.000000     1.000000     1.000000
```

```
#Converting the Median Income column to numeric type
fin_df['Med_Income'] = pd.to_numeric(fin_df.Med_Income)
```

```
# Checks what all columns in the dataframe contain only numeric values
fin_df.apply(lambda s: pd.to_numeric(s,
errors='coerce').notnull().all())
```

```
State                False
AreaName             False
All_Poverty           True
```

```
M_Poverty              True
F_Poverty              True
FIPS                   True
Med_Income            False
M_With                 True
M_Without              True
F_With                 True
F_Without              True
All_With               True
All_Without            True
Avg_Ann_Incidence     False
recent_trend          False
Avg_Ann_Deaths        False
Rising                 True
Falling                True
dtype: bool
```

We wont be doing anything with the null values we have created as out library Statsmodels can handle missing data !

Visualization

```
# Scatter plot Avg_Ann_Incidence vs Avg_Ann_Deaths
fin_df.plot(x = 'Avg_Ann_Incidence', y = 'Avg_Ann_Deaths', kind=
'scatter', s=0.1, xlim=[0, 600], ylim=[0, 600], title = 'Scatter
Plot')
```

```
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```

```
<AxesSubplot:title={'center':'Scatter Plot'},
xlabel='Avg_Ann_Incidence', ylabel='Avg_Ann_Deaths'>
```

```
# Pairplot
sns.set(style='whitegrid', context='notebook')
sns.pairplot(fin_df[['All_Poverty', 'M_Poverty', 'F_Poverty',
'Med_Income']], height=2)
```

```
<seaborn.axisgrid.PairGrid at 0x1b2f47ab688>
```

```python
# Correlation Heatmap
cols = ['All_Poverty', 'M_Poverty', 'F_Poverty', 'Med_Income']
cm = fin_df[cols].corr()
sns.set(font_scale=1.5)
hm = sns.heatmap(cm, cbar=True, annot = True, square=True, fmt='.2f',
annot_kws={'size':15},
                 yticklabels=cols, xticklabels=cols)
```

We see that the columns All_Poverty, M and F poverty are very correlated and hence we need to drop two from these !!

```
fin_df.drop(['M_Poverty', 'F_Poverty'], axis=1, inplace=True)

cols = ['M_With', 'M_Without', 'F_With', 'F_Without', 'All_With',
        'All_Without']
cm = fin_df[cols].corr()
sns.set(font_scale=1.5)
hm = sns.heatmap(cm, cbar=True, annot = True, square=True, fmt='.2f',
annot_kws={'size':15},
                 yticklabels=cols, xticklabels=cols)
```

```
fin_df.drop(['M_With', 'M_Without', 'F_With', 'F_Without'], axis=1,
inplace=True)

sns.set(style='whitegrid', context='notebook')
sns.pairplot(fin_df[['All_Poverty', 'Med_Income', 'All_With',
        'All_Without']], height=2)
```

```
<seaborn.axisgrid.PairGrid at 0x1b2f7688488>
```

```
cols = ['All_Poverty', 'Med_Income', 'All_With','All_Without']
cm = fin_df[cols].corr()
sns.set(font_scale=1.5)
hm = sns.heatmap(cm, cbar=True, annot = True, square=True, fmt='.2f',
annot_kws={'size':15},
                 yticklabels=cols, xticklabels=cols)
```

```
# Scatter Plot
def visualize_scatter_pov(col):
    fig1 = plt.figure(figsize = (14,10))

    ax3 = fig1.add_subplot(223)
    fin_df.plot(x = 'Avg_Ann_Deaths', y = col, kind= 'scatter', s=0.1,
xlim = [0, 200], ylim = [0, 40000], ax = ax3)
    ax4 = fig1.add_subplot(224)
    fin_df.plot(x = 'Avg_Ann_Incidence', y = col, kind= 'scatter',
s=0.1,xlim = [0, 200], ylim = [0, 40000], ax = ax4)

visualize_scatter_pov('All_Poverty')
```

```
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```

```
def visualize_scatter_inc(col):
    fig1 = plt.figure(figsize = (14,10))
    ax3 = fig1.add_subplot(223)
    fin_df.plot(x = 'Avg_Ann_Deaths', y = col, kind= 'scatter',xlim =
[0, 200], ylim = [0, 100000], s=0.1, ax = ax3)
    ax4 = fig1.add_subplot(224)
    fin_df.plot(x = 'Avg_Ann_Incidence', y = col, kind= 'scatter',
s=0.1,xlim = [0, 200], ylim = [0, 100000], ax = ax4)

visualize_scatter_inc('Med_Income')
```

```
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```

```
def visualize_scatter_with(col):
    fig1 = plt.figure(figsize = (14,10))
    ax3 = fig1.add_subplot(223)
    fin_df.plot(x = 'Avg_Ann_Deaths', y = col, kind= 'scatter',xlim =
[0, 100], ylim = [0, 100000], s=0.1, ax = ax3)
    ax4 = fig1.add_subplot(224)
    fin_df.plot(x = 'Avg_Ann_Incidence', y = col, kind= 'scatter',
s=0.1,xlim = [0, 100], ylim = [0, 100000], ax = ax4)

visualize_scatter_with('All_With')
```

```
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```

```
def visualize_scatter_without(col):
    fig1 = plt.figure(figsize = (14,10))
    ax3 = fig1.add_subplot(223)
    fin_df.plot(x = 'Avg_Ann_Deaths', y = col, kind= 'scatter',xlim =
[0, 100], ylim = [0, 20000], s=0.1, ax = ax3)
    ax4 = fig1.add_subplot(224)
    fin_df.plot(x = 'Avg_Ann_Incidence', y = col, kind= 'scatter',
s=0.1,xlim = [0, 100], ylim = [0, 20000], ax = ax4)

visualize_scatter_without('All_Without')
```

```
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
*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 2D array with a single row if you intend to
specify the same RGB or RGBA value for all points.
```

Statistical Linear Regression Modelling using Statsmodel Library

Unlike SKLearn, statsmodels doesn't automatically fit a constant, so you need to use the method sm.add_constant(X) in order to add a constant. Adding a constant, while not necessary, makes your line fit much better. For example, if you have a line with an intercept of -2000 and you try to fit the same line through the origin, you're going to get an inferior line. Once we add a constant (or an intercept if you're thinking in line terms), you'll see that the coefficients are the same in SKLearn and statsmodels.

```
fin_df.columns
```

```
Index(['State', 'AreaName', 'All_Poverty', 'FIPS', 'Med_Income',
'All_With',
       'All_Without', 'Avg_Ann_Incidence', 'recent_trend',
'Avg_Ann_Deaths',
       'Rising', 'Falling'],
      dtype='object')
```

```
res = ''
for i in fin_df.columns:
    res += str(i) + ' + '
print(res)
```

```
State + AreaName + All_Poverty + FIPS + Med_Income + All_With +
All_Without + Avg_Ann_Incidence + recent_trend + Avg_Ann_Deaths +
Rising + Falling +
```

```
import statsmodels.formula.api as smf

# Using statmodels library for Linear Regression Modelling
model1 = smf.ols(formula='Avg_Ann_Incidence ~ All_Poverty + Med_Income
+ All_With + All_Without + Rising + Falling', data=fin_df).fit()
print(model1.summary())
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:      Avg_Ann_Incidence   R-squared:
0.922
Model:                            OLS   Adj. R-squared:
0.921
Method:                 Least Squares   F-statistic:
5707.
Date:                Mon, 06 Dec 2021   Prob (F-statistic):
0.00
Time:                        23:09:39   Log-Likelihood:
-15493.
No. Observations:                2924   AIC:
3.100e+04
Df Residuals:                    2917   BIC:
3.104e+04
Df Model:                           6

Covariance Type:            nonrobust

==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
```

```
Intercept      36.8125      3.931      9.365      0.000      29.105
44.520
All_Poverty     0.0005   8.38e-05      6.497      0.000       0.000
0.001
Med_Income     -0.0005   8.48e-05     -5.914      0.000      -0.001
-0.000
All_With        0.0008   1.36e-05     55.181      0.000       0.001
0.001
All_Without    -0.0014   6.39e-05    -22.047      0.000      -0.002
-0.001
Rising         -6.5225      7.452     -0.875      0.381     -21.134
8.089
Falling        11.5051      3.775      3.047      0.002       4.102
18.908
==============================================================================
Omnibus:                     1643.837   Durbin-Watson:
1.614
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
599626.270
Skew:                           1.406   Prob(JB):
0.00
Kurtosis:                      73.098   Cond. No.
2.50e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.5e+06. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```python
# Fitting to Average Deaths
model2 = smf.ols(formula='Avg_Ann_Deaths ~ All_Poverty + Med_Income  +
All_With + All_Without + Rising + Falling', data=fin_df).fit()
print(model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         Avg_Ann_Deaths   R-squared:
0.925
Model:                            OLS   Adj. R-squared:
0.924
Method:                 Least Squares   F-statistic:
5728.
Date:                Mon, 06 Dec 2021   Prob (F-statistic):
```

```
0.00
Time:                        23:09:39   Log-Likelihood:
-13977.
No. Observations:                2809   AIC:
2.797e+04
Df Residuals:                    2802   BIC:
2.801e+04
Df Model:                           6

Covariance Type:            nonrobust

==================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept        32.1884      2.948     10.919      0.000      26.408      37.969
All_Poverty       0.0004   6.07e-05      7.192      0.000       0.000       0.001
Med_Income       -0.0004   6.35e-05     -6.973      0.000      -0.001      -0.000
All_With          0.0005   9.89e-06     51.334      0.000       0.000       0.001
All_Without      -0.0009   4.58e-05    -18.761      0.000      -0.001      -0.001
Rising           -6.4509      5.664     -1.139      0.255     -17.557       4.655
Falling           8.1574      2.736      2.982      0.003       2.793      13.522
==================================================================================
Omnibus:                     1667.512   Durbin-Watson:
1.612
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
293161.245
Skew:                           1.752   Prob(JB):
0.00
Kurtosis:                      52.925   Cond. No.
2.65e+06
==================================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.65e+06. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

Multicollinearity

```
# Importing VIF to check for multicollinearity
from statsmodels.stats.outliers_influence import
variance_inflation_factor

X = fin_df[[ 'All_Poverty',  'Med_Income', 'All_With', 'All_Without',
'Rising', 'Falling', 'Avg_Ann_Incidence']]

X.columns
```

```
Index(['All_Poverty', 'Med_Income', 'All_With', 'All_Without',
'Rising',
       'Falling', 'Avg_Ann_Incidence'],
      dtype='object')
```

```
X.dropna(inplace=True)
```

```
C:\Users\hp\Anaconda3\lib\site-packages\pandas\util\
_decorators.py:311: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  return func(*args, **kwargs)
```

```
X1 = X[['All_Poverty', 'Med_Income', 'All_With', 'All_Without',
'Rising',
       'Falling']]

# Getting the VIFs corresponding to each of the features and then
treating the one with the highest value
pd.DataFrame([[var, variance_inflation_factor(X1.values,
X1.columns.get_loc(var))] for var in X1.columns],
                    index=range(X1.shape[1]), columns=['Variable',
'VIF'])
```

```
      Variable        VIF
0  All_Poverty  25.654166
1   Med_Income   1.271954
2     All_With  15.627624
3  All_Without  16.031002
4       Rising   1.015347
5      Falling   1.201154
```

We can see that the model suffers from high multicollinearity as the VIF is high for the variable and thus might lead to coefficients that are statistically insignificant !!

```
X.drop('All_Poverty', axis = 1, inplace=True)
X1.drop('All_Poverty', axis = 1, inplace=True)
```

```
C:\Users\hp\Anaconda3\lib\site-packages\pandas\core\frame.py:4913:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  errors=errors,
```

```
pd.DataFrame([[var, variance_inflation_factor(X1.values,
X1.columns.get_loc(var))] for var in X1.columns],
                   index=range(X1.shape[1]), columns=['Variable',
'VIF'])
```

```
      Variable       VIF
0   Med_Income  1.265045
1     All_With  9.776656
2  All_Without  8.926300
3       Rising  1.015175
4      Falling  1.199055
```

```
X.drop('All_Without', axis = 1, inplace=True)
X1.drop('All_Without', axis = 1, inplace=True)
```

```
C:\Users\hp\Anaconda3\lib\site-packages\pandas\core\frame.py:4913:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  errors=errors,
```

```
pd.DataFrame([[var, variance_inflation_factor(X1.values,
X1.columns.get_loc(var))] for var in X1.columns],
                   index=range(X1.shape[1]), columns=['Variable',
'VIF'])
```

```
     Variable       VIF
0  Med_Income  1.207631
1    All_With  1.257105
2      Rising  1.015051
3     Falling  1.193283
```

```
# Final model after treating for multicollinearity
model1 = smf.ols(formula='Avg_Ann_Incidence ~Med_Income  + All_With +
Rising + Falling', data=fin_df).fit()
print(model1.summary())
```

```
                            OLS Regression Results
```

```
==============================================================================
Dep. Variable:      Avg_Ann_Incidence   R-squared:                       0.906
Model:                            OLS   Adj. R-squared:                  0.906
Method:                 Least Squares   F-statistic:                     7020.
Date:                Mon, 06 Dec 2021   Prob (F-statistic):               0.00
Time:                        23:09:39   Log-Likelihood:                -15759.
No. Observations:                2924   AIC:                         3.153e+04
Df Residuals:                    2919   BIC:                         3.156e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     24.4731      3.875      6.315      0.000      16.874      32.072
Med_Income    -0.0002   8.23e-05     -1.898      0.058      -0.000    5.17e-06
All_With       0.0006   3.84e-06    154.332      0.000       0.001       0.001
Rising        -6.8821      8.159     -0.844      0.399     -22.879       9.115
Falling       16.1143      4.122      3.909      0.000       8.032      24.196
==============================================================================
Omnibus:                     1260.365   Durbin-Watson:                   1.646
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1570186.457
Skew:                           0.370   Prob(JB):                         0.00
Kurtosis:                     116.523   Cond. No.                     2.42e+06
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.42e+06. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

Detecting Outliers

```
fin_df.columns
```

```
Index(['State', 'AreaName', 'All_Poverty', 'FIPS', 'Med_Income',
'All_With',
       'All_Without', 'Avg_Ann_Incidence', 'recent_trend',
'Avg_Ann_Deaths',
       'Rising', 'Falling'],
      dtype='object')
```

```
fin_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3134 entries, 0 to 3133
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   State              3134 non-null   object
 1   AreaName           3134 non-null   object
 2   All_Poverty        3134 non-null   int64
 3   FIPS               3134 non-null   int64
 4   Med_Income         3133 non-null   float64
 5   All_With           3134 non-null   int64
 6   All_Without        3134 non-null   int64
 7   Avg_Ann_Incidence  2925 non-null   float64
 8   recent_trend       3134 non-null   object
 9   Avg_Ann_Deaths     2809 non-null   float64
 10  Rising             3134 non-null   int64
 11  Falling            3134 non-null   int64
dtypes: float64(3), int64(6), object(3)
memory usage: 382.8+ KB
```

```
data_mod1 = fin_df.copy()
#using IQR method
dat = data_mod1['Med_Income']
Q1 = dat.quantile(0.25)
Q3 = dat.quantile(0.75)
IQR = Q3 - Q1    #IQR is interquartile range.

filter = (dat >= Q1 - 3 * IQR) & (dat <= Q3 + 3 *IQR)
data_mod1 = data_mod1.loc[filter]
print(data_mod1.shape)
```

```
(3112, 12)

# Boxplot for the initial data
ax = sns.boxplot(data=fin_df, orient="h", palette="Set2")
```

```
# Fitting to Average Deaths
model2 = smf.ols(formula='Avg_Ann_Deaths ~ All_Poverty + Med_Income  +
All_With + All_Without + Rising + Falling', data=data_mod1).fit()
print(model2.summary())

                            OLS Regression Results

==============================================================================
========
Dep. Variable:          Avg_Ann_Deaths   R-squared:
0.928
Model:                             OLS   Adj. R-squared:
0.927
Method:                  Least Squares   F-statistic:
5933.
Date:                 Mon, 06 Dec 2021   Prob (F-statistic):
0.00
Time:                         23:09:40   Log-Likelihood:
-13809.
No. Observations:                 2788   AIC:
2.763e+04
Df Residuals:                     2781   BIC:
2.767e+04
Df Model:                            6
```

```
Covariance Type:            nonrobust
==========================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------
---------
Intercept       30.6782      3.056     10.040      0.000      24.687
36.670
All_Poverty      0.0003   6.18e-05      4.587      0.000       0.000
0.000
Med_Income      -0.0004   6.63e-05     -6.240      0.000      -0.001
-0.000
All_With         0.0005   1.03e-05     52.225      0.000       0.001
0.001
All_Without     -0.0008   4.49e-05    -18.758      0.000      -0.001
-0.001
Rising          -6.4949      5.536     -1.173      0.241     -17.349
4.360
Falling          7.2932      2.718      2.683      0.007       1.964
12.623
==========================================================================
========
Omnibus:                     1780.288   Durbin-Watson:
1.606
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
313764.257
Skew:                           1.992   Prob(JB):
0.00
Kurtosis:                      54.818   Cond. No.
2.62e+06
==========================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.62e+06. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```
data_mod1 = fin_df.copy()
for i in ['All_Poverty', 'Med_Income', 'All_With', 'All_Without',
'Rising', 'Falling']:
    #using IQR method
    dat = data_mod1[i]
    Q1 = dat.quantile(0.25)
    Q3 = dat.quantile(0.75)
```

```
    IQR = Q3 - Q1    #IQR is interquartile range.

    filter = (dat >= Q1 - 3 * IQR) & (dat <= Q3 + 3 *IQR)
    data_mod1 = data_mod1.loc[filter]
    print(data_mod1.shape)
```

```
(2563, 12)
(2563, 12)
(2559, 12)
(2553, 12)
(2553, 12)
(2455, 12)
```

```
# Boxplot for the filtered data
ax = sns.boxplot(data=data_mod1[['All_Poverty', 'Med_Income',
'All_With', 'All_Without', 'Rising', 'Falling']], orient="h",
palette="Set2")
```

```
fin_df = data_mod1

# Fitting to Average Deaths
model1 = smf.ols(formula='Avg_Ann_Deaths ~ All_Poverty + Med_Income  +
All_With + All_Without + Rising + Falling', data=data_mod1).fit()
```

Normality of Errors

```
# histogram superimposed by normal curve
plt.figure(figsize=(10,6))
```

```
import scipy.stats as stats
mu = np.mean(model1.resid)
sigma = np.std(model1.resid)
pdf = stats.norm.pdf(sorted(model1.resid), mu, sigma)
pdf2 = stats.t.pdf(sorted(model1.resid), df = 1, loc = mu,
scale=sigma,)
plt.xlabel('x')
plt.ylabel('Probability Density')
plt.hist(model1.resid, bins=100, density= True)
plt.plot(sorted(model1.resid), pdf, color='r', linewidth=2, label =
'Gaussian')
plt.plot(sorted(model1.resid), pdf2, color='g', linewidth=2, label =
't distribution')
plt.legend()
plt.show()
```

```
import statsmodels.graphics.gofplots as sm

# QQplot
fig, [ax1, ax2] = plt.subplots(1,2, figsize=(10,3))
sm.qqplot(model1.resid, stats.norm, fit=True, line='45', ax=ax1)
ax1.set_title("normal distribution")
sm.qqplot(model1.resid, stats.t, fit=True, line='45', ax = ax2)
ax2.set_title("t distribution")

plt.show()
```

As we can see that the QQ plot for the normal distribution is not close to the straight line at the ends and it deviates to the top at the right and to the bottom at the left, we can say that the tails for the residuals are heavier than a normal distribution, hence on following that I tried using a t distribution, which gives a much better QQ plot and hence we can confirm that the residuals come from the t distribution !! Fatter tails suggest we have more number of outliers !

```
result = model1

# plot actual values versus residuals
plt.figure(figsize=(14,7))
plt.title('Scatter plot of Residuals vs Fitted Values')
plt.scatter(y=result.resid, x=result.fittedvalues, alpha=0.5, s=22)
plt.axhline(y=0, color='r', linestyle="--", alpha=0.5)
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.xlim(-10, 100)
plt.ylim(-200, 200)
plt.show()
```

We can see a cone shaped residual plot which is common in cases of heteroscadasticity wherein when the fitted value increases, the variance also increases. Hence our model suffers from high heteroscadasticity.

Heteroscedasticity is a systematic change in the spread of the residuals over the range of measured values. Heteroscedasticity is a problem because ordinary least squares (OLS) regression assumes that all residuals are drawn from a population that has a constant variance (homoscedasticity). Heteroscedasticity, also spelled heteroskedasticity, occurs more often in datasets that have a large range between the largest and smallest observed values. While there are numerous reasons why heteroscedasticity can exist, a common explanation is that the error variance changes proportionally with a factor. This factor might be a variable in the model.

While heteroscedasticity does not cause bias in the coefficient estimates, it does make them less precise. Lower precision increases the likelihood that the coefficient estimates are further from the correct population value. Heteroscedasticity tends to produce p-values that are smaller than they should be. This effect occurs because heteroscedasticity increases the variance of the coefficient estimates but the OLS procedure does not detect this increase. Consequently, OLS calculates the t-values and F-values using an underestimated amount of variance. This problem can lead you to conclude that a model term is statistically significant when it is actually not significant.

```python
# Importing the required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')

import seaborn as sns

# Reading the data
df = pd.read_csv('train.csv')

df_test = pd.read_csv('test.csv')

df.head()
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age
SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
2                             Heikkinen, Miss. Laina  female  26.0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                           Allen, Mr. William Henry    male  35.0
0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
```

```
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df.describe().transpose()
```

```
             count        mean         std   min       25%       50%
75%  \
PassengerId  891.0  446.000000  257.353842  1.00  223.5000  446.0000
668.5
Survived     891.0    0.383838    0.486592  0.00    0.0000    0.0000
1.0
Pclass       891.0    2.308642    0.836071  1.00    2.0000    3.0000
3.0
Age          714.0   29.699118   14.526497  0.42   20.1250   28.0000
38.0
SibSp        891.0    0.523008    1.102743  0.00    0.0000    0.0000
1.0
Parch        891.0    0.381594    0.806057  0.00    0.0000    0.0000
0.0
Fare         891.0   32.204208   49.693429  0.00    7.9104   14.4542
31.0

                  max
PassengerId  891.0000
Survived       1.0000
Pclass         3.0000
Age           80.0000
SibSp          8.0000
Parch          6.0000
Fare         512.3292
```

```
f,ax=plt.subplots(1,2,figsize=(18,8))
df['Survived'].value_counts().plot.pie(ax = ax[0],
explode=[0,0.1],autopct='%1.1f%%',shadow=True)
ax[0].set_title('Survived')
ax[0].set_ylabel('')
sns.countplot('Survived',data=df,ax=ax[1])
ax[1].set_title('Survived')
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
Text(0.5, 1.0, 'Survived')
```

Overall Probability of Survival ~ 38%

Pre-processing, EDA and Feature Generation

Handling Missing Values

Handling Age First

```
df['Age'].describe()
```

```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

```
df.corr()
```

```
             PassengerId  Survived    Pclass       Age     SibSp
Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036847 -0.057527 -
0.001652
Survived       -0.005007  1.000000 -0.338481 -0.077221 -0.035322
0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.369226  0.083081
0.018443
Age             0.036847 -0.077221 -0.369226  1.000000 -0.308247 -
0.189119
SibSp          -0.057527 -0.035322  0.083081 -0.308247  1.000000
0.414838
Parch          -0.001652  0.081629  0.018443 -0.189119  0.414838
1.000000
Fare            0.012658  0.257307 -0.549500  0.096067  0.159651
0.216225

                  Fare
PassengerId  0.012658
Survived     0.257307
Pclass      -0.549500
Age          0.096067
SibSp        0.159651
Parch        0.216225
Fare         1.000000
```

We see that the Pclass column has the maximum absolute correlation with our target class Survived.

```
df.corr()['Pclass'].sort_values()
```

```
Fare          -0.549500
Age           -0.369226
Survived      -0.338481
PassengerId   -0.035144
Parch          0.018443
SibSp          0.083081
Pclass         1.000000
Name: Pclass, dtype: float64
```

```
age_by_pclass_sex= df.groupby(['Sex','Pclass']).median()['Age']
```

```
age_by_pclass_sex
```

```
Sex     Pclass
female  1         35.0
        2         28.0
        3         21.5
male    1         40.0
        2         30.0
```

```
        3         25.0
Name: Age, dtype: float64
```

```
median_all = df['Age'].median()
median_all
```

```
28.0
```

```
df['Age'] = df.groupby(['Sex', 'Pclass'])['Age'].apply(lambda
x:x.fillna(x.median()))
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Let's jump to Embarked

```
df['Embarked'].describe()
```

```
count     889
unique      3
top         S
freq      644
Name: Embarked, dtype: object
```

only two values are missing!

```
df[df['Embarked'].isnull()]
```

```
     PassengerId  Survived  Pclass
Name  \
61            62         1       1                          Icard, Miss.
```

```
Amelie
829          830         1       1  Stone, Mrs. George Nelson (Martha
Evelyn)

        Sex   Age  SibSp  Parch  Ticket  Fare Cabin Embarked
61   female  38.0      0      0  113572  80.0   B28      NaN
829  female  62.0      0      0  113572  80.0   B28      NaN
```

Let's Handle Cabin

We have (891-204) = 687 missing Cabin values

```
df['Cabin'].unique()
```

```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60',
'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49',
'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91',
'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10',
'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63',
'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F
G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46',
'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17',
'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

We can see that there are many missing values thus we replace the missing values by a new class M (missing), plus we have cabins with different classes followed by numbers, thus we can just keep the classes.

```
df['Cabin']=df['Cabin'].fillna('M')

df['Cabin']=df['Cabin'].apply(lambda str: str[0])
```

```
df.loc[df[df['Cabin']=='T'].index,'Cabin']='A'

g = sns.countplot(x='Cabin',data=df,hue='Pclass', )
plt.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x1fb148e3848>
```

```
sns.countplot(x='Cabin',data=df,hue='Survived')
```

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

On visualizing the final Cabin column we see that most of the people in the missing class belong to the Pclass 3 and have little chance of survival. We see a higher chance of survival in most of the other cabin classes, with the least being in cabin A and the highest chance being in the cabin class B. We also observe that most of the people in the classes C, E, G, D, A, B belong ot the Pclass 1, and have high chances of survival, thus indicating that Pclass might be a factor resulting in higher chances of survival.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        891 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

EDA and Feature Generation

```
df['Embarked'].unique()
```

```
array(['S', 'C', 'Q', nan], dtype=object)
```

```
sns.countplot(x='Embarked',data=df,hue='Survived')
```

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

```
f,ax=plt.subplots(2,2,figsize=(20,15))
sns.countplot('Embarked',data=df,ax=ax[0,0])
ax[0,0].set_title('No. Of Passengers Boarded')
sns.countplot('Embarked',hue='Sex',data=df,ax=ax[0,1])
ax[0,1].set_title('Male-Female Split for Embarked')
sns.countplot('Embarked',hue='Survived',data=df,ax=ax[1,0])
ax[1,0].set_title('Embarked vs Survived')
sns.countplot('Embarked',hue='Pclass',data=df,ax=ax[1,1])
ax[1,1].set_title('Embarked vs Pclass')
plt.subplots_adjust(wspace=0.2,hspace=0.5)
plt.show()
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
```

```
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

1)Maximum passenegers boarded from S. Majority of them being from Pclass3.

2)The Passengers from C look to be lucky as a good proportion of them survived. The reason for this maybe the rescue of all the Pclass1 and Pclass2 Passengers.

3)The Embark S looks to the port from where majority of the rich people boarded. Still the chances for survival is low here, that is because many passengers from Pclass3 around 81% didn't survive.

4)Port Q had almost 95% of the passengers were from Pclass3.

```
df['Ticket_Frequency']=df.groupby('Ticket')
['Ticket'].transform('count')

sns.countplot(x='Ticket_Frequency',data=df,hue='Survived')

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

On careful observation we see that different ticket frequency has different rates of survival with the highest being for the group of two and the chance for groups of more than 4 being very less.

```
df['Title']=df['Name'].str.split(', ',expand=True)
[1].str.split('.',expand=True)[0]

df['Title'].unique()

array(['Mr', 'Mrs', 'Miss', 'Master', 'Don', 'Rev', 'Dr', 'Mme', 'Ms',
       'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'the Countess',
       'Jonkheer'], dtype=object)

sns.countplot(x='Title',data=df,hue='Survived')

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

```
df['IsMarried']=0

df['IsMarried'].loc[df['Title']=='Mrs']=1

C:\Users\hp\Anaconda3\lib\site-packages\pandas\core\indexing.py:1732:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)

df['IsMarried'].unique()

array([0, 1], dtype=int64)

sns.countplot(x='IsMarried',data=df,hue='Survived')

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

We created a new feature called IsMarried that is checked for title Mrs. We see that being married has a greater chance of survival !

```
df['Title'] = df['Title'].replace(['Miss', 'Mrs','Ms', 'Mlle', 'Lady',
'Mme', 'the Countess', 'Dona'], 'Miss_Mrs_Ms')

df['Title'] = df['Title'].replace(['Dr', 'Col', 'Major', 'Jonkheer',
'Capt', 'Sir', 'Don', 'Rev'], 'Others')

sns.countplot(x='Title',data=df,hue='Survived')

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

```
sns.boxplot(x='Title', y='Age', data=df)
```

```
<AxesSubplot:xlabel='Title', ylabel='Age'>
```

1. We converted Title into 4 groups.
2. We can see most of the people are from Mr. grp where the probability of surviving is low while the probability of Miss_Mrs_Ms and Master is higher for surviving.
3. We have very less people belonging to the other classes.

```
df['Child']=0

df['Child'].loc[df['Age']<18]=1
```

```
C:\Users\hp\Anaconda3\lib\site-packages\pandas\core\indexing.py:1732:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

```
df['Child'].unique()
```

```
array([0, 1], dtype=int64)
```

```
sns.countplot(x='Child',data=df,hue='Survived')
```

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

We created a Child feature for people below 18 years of age, now we can see that children have a higher chance of getting out !

We create a new feature Family size, which is essentialy No. of Parents + Chilren + Siblings + Spouse + 1

```
df['Fam_Size']=df['Parch']+df['SibSp']+1

sns.countplot(x='Fam_Size',data=df,hue='Survived')
```

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

```
df.loc[:,'FsizeD']='Alone'
df.loc[(df['Fam_Size']>1),'FsizeD']='Small'
df.loc[(df['Fam_Size']>4),'FsizeD']='Big'
```

Now we create three different categories of Family Size

```
sns.countplot(x='FsizeD',data=df,hue='Survived')
<AxesSubplot:xlabel='FsizeD', ylabel='count'>
```

We see that people with small family size have the highest chance of Surviving, and the ones with a big family, which the lease chance !

```
df=df.drop(['Name','Ticket','Fam_Size'],axis=1)

df=df.drop(['PassengerId'],axis=1)
```

Visualization

```
sns.countplot(x='Pclass',data=df,hue='Survived')

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

We see that the people belonging to the Pclass 1 have a higher chance of survival, which keeps on decreasing as we go down the classes !

```
sns.countplot(x='Sex',data=df,hue='Survived')
```

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

1. Proportion of male and female: ~2/3 vs ~1/3

2. Male is much less likely to survive, with only 20% chance of survival. For female, >70% chance of survival.
3. Obviously, Sex is an important feature to predict survival.

```
plt.hist(df.Age, edgecolor="black")
plt.xlabel('Age')
plt.ylabel('count')
plt.show()
```

```
sns.boxplot(x='Survived', y='Age', data=df)
plt.show()
```

1. Passengers are mainly aged 20–40.
2. Younger passengers tends to survive.

```
sns.countplot(x='SibSp',data=df,hue='Survived')
```

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

1. Most of the passengers are travelling alone
2. The ones with 1 sibling/spouse are more likely to survive

```
sns.countplot(x='Parch',data=df,hue='Survived')
```

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

```
sns.barplot(x='Parch', y='Survived', data=df)
plt.show()
```

1. ~ >70% passengers travel without parents/children.

2. Passengers travelling with parents/children are more likely to survive than those not.

```
sns.distplot(df.Fare)
plt.show()
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
sns.boxplot(x='Survived', y='Fare', data=df)
plt.show()
```

1. We see that there are people paying too much for their tickets (outliers)
2. We see that people with higher fares are likely to survive !

```
sns.countplot(x='Embarked',data=df,hue='Survived')
```

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

1. We see most of the people were embarked from the port S
2. People embarked from the C port are more likely to survive !

```
df['Age_band']=0
df.loc[df['Age']<=16,'Age_band']=0
df.loc[(df['Age']>16)&(df['Age']<=32),'Age_band']=1
df.loc[(df['Age']>32)&(df['Age']<=48),'Age_band']=2
df.loc[(df['Age']>48)&(df['Age']<=64),'Age_band']=3
df.loc[df['Age']>64,'Age_band']=4

df['Fare_Range']=pd.qcut(df['Fare'],4)

cm = df.corr()
sns.set(font_scale=1.5, )
hm = sns.heatmap(cm, cbar=True, annot = True, square=True, fmt='.2f',
annot_kws={'size':15})
      #          yticklabels=cols, xticklabels=cols)

fig = plt.gcf()
fig.set_size_inches(10,8)
```

Converting into Dummies

```
dummies = pd.get_dummies(df['Embarked'],drop_first=True)

df = pd.concat([df.drop('Embarked',axis=1),dummies],axis=1)

dummies = pd.get_dummies(df['Cabin'],drop_first=True)

df = pd.concat([df.drop('Cabin',axis=1),dummies],axis=1)

dummies = pd.get_dummies(df['Sex'],drop_first=True)

df = pd.concat([df.drop('Sex',axis=1),dummies],axis=1)

dummies = pd.get_dummies(df['Title'],drop_first=True)

df = pd.concat([df.drop('Title',axis=1),dummies],axis=1)

dummies = pd.get_dummies(df['FsizeD'],drop_first=True)

df = pd.concat([df.drop('FsizeD',axis=1),dummies],axis=1)

df.head()
```

```
   Survived  Pclass   Age  SibSp  Parch     Fare  Ticket_Frequency
IsMarried  \
0         0       3  22.0      1      0   7.2500                 1
0
1         1       1  38.0      1      0  71.2833                 1
1
2         1       3  26.0      0      0   7.9250                 1
0
3         1       1  35.0      1      0  53.1000                 2
1
4         0       3  35.0      0      0   8.0500                 1
0

   Child  Age_band  ...  E  F  G  M  male  Miss_Mrs_Ms  Mr  Others
Big  Small
0      0         1  ...  0  0  0  1     1            0   1       0
0      1
1      0         2  ...  0  0  0  0     0            1   0       0
0      1
2      0         1  ...  0  0  0  1     0            1   0       0
0      0
3      0         2  ...  0  0  0  0     0            1   0       0
0      1
4      0         2  ...  0  0  0  1     1            0   1       0
0      0

[5 rows x 26 columns]
```

```
df.corr()['Survived'].sort_values().drop('Survived').plot(kind='bar')
```

```
<AxesSubplot:>
```

```
df.isnull().sum()
```

```
Survived             0
Pclass               0
Age                  0
SibSp                0
Parch                0
Fare                 0
Ticket_Frequency     0
IsMarried            0
Child                0
Age_band             0
Fare_Range           0
Q                    0
S                    0
```

```
B                    0
C                    0
D                    0
E                    0
F                    0
G                    0
M                    0
male                 0
Miss_Mrs_Ms          0
Mr                   0
Others               0
Big                  0
Small                0
dtype: int64
```

```
df.drop('Age', axis = 1, inplace=True)
```

Statistical Logistic Regression Modelling using Statsmodel Library

```
from sklearn.metrics import confusion_matrix, f1_score
```

```
df.columns
```

```
Index(['Survived', 'Pclass', 'SibSp', 'Parch', 'Fare',
'Ticket_Frequency',
       'IsMarried', 'Child', 'Age_band', 'Fare_Range', 'Q', 'S', 'B',
'C', 'D',
       'E', 'F', 'G', 'M', 'male', 'Miss_Mrs_Ms', 'Mr', 'Others',
'Big',
       'Small'],
      dtype='object')
```

```
res = ''
for i in df.columns:
    res += str(i) + ' + '
print(res)
```

```
Survived + Pclass + SibSp + Parch + Fare + Ticket_Frequency +
IsMarried + Child + Age_band + Fare_Range + Q + S + B + C + D + E + F
+ G + M + male + Miss_Mrs_Ms + Mr + Others + Big + Small +
```

```
df.head()
```

```
   Survived  Pclass  SibSp  Parch     Fare  Ticket_Frequency
IsMarried  \
0         0       3      1      0   7.2500                 1
0
1         1       1      1      0  71.2833                 1
1
```

```
2         1       3      0      0   7.9250                 1
0
3         1       1      1      0  53.1000                 2
1
4         0       3      0      0   8.0500                 1
0

   Child  Age_band       Fare_Range  ...  E  F  G  M  male
Miss_Mrs_Ms  Mr  \
0      0         1   (-0.001, 7.91]  ...  0  0  0  1     1
0   1
1      0         2  (31.0, 512.329]  ...  0  0  0  0     0
1   0
2      0         1   (7.91, 14.454]  ...  0  0  0  1     0
1   0
3      0         2  (31.0, 512.329]  ...  0  0  0  0     0
1   0
4      0         2   (7.91, 14.454]  ...  0  0  0  1     1
0   1

   Others  Big  Small
0       0    0      1
1       0    0      1
2       0    0      0
3       0    0      1
4       0    0      0

[5 rows x 25 columns]
```

```
import statsmodels.formula.api as smf

m1 = smf.logit(
  formula='Survived ~ Pclass + SibSp + Parch + Ticket_Frequency +
IsMarried + Child + Age_band + Fare_Range + Q + S + B + C + D + E + F
+ G + M + male + Mr + Others + Big + Small',
  data=df) \
.fit()

m1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.393608
         Iterations 7

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

==========================================================================
========
Dep. Variable:                Survived   No. Observations:
```

```
891
Model:                           Logit   Df Residuals:
866
Method:                            MLE   Df Model:
24
Date:                Wed, 01 Dec 2021   Pseudo R-squ.:
0.4089
Time:                        05:55:35   Log-Likelihood:
-350.70
converged:                       True   LL-Null:
-593.33
Covariance Type:            nonrobust   LLR p-value:
1.920e-87
==========================================================================
=====================================================
                                                                  coef
std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
-----------------------------------------------------
Intercept                                                       3.9991
0.931      4.297      0.000       2.175       5.823
Fare_Range[T.Interval(7.91, 14.454, closed='right')]      0.1753
0.318      0.552      0.581      -0.448       0.798
Fare_Range[T.Interval(14.454, 31.0, closed='right')]      0.4089
0.393      1.041      0.298      -0.361       1.179
Fare_Range[T.Interval(31.0, 512.329, closed='right')]     0.5037
0.542      0.929      0.353      -0.559       1.567
Pclass                                                         -0.7701
0.229     -3.357      0.001      -1.220      -0.320
SibSp                                                          -0.1094
0.227     -0.483      0.629      -0.554       0.335
Parch                                                           0.0651
0.228      0.285      0.775      -0.382       0.512
Ticket_Frequency                                                0.0699
0.129      0.543      0.587      -0.182       0.322
IsMarried                                                       0.5934
0.378      1.569      0.117      -0.148       1.335
Child                                                           0.0872
0.417      0.209      0.834      -0.729       0.904
Age_band                                                       -0.4638
0.174     -2.671      0.008      -0.804      -0.124
Q                                                              -0.0446
0.420     -0.106      0.915      -0.868       0.779
S                                                              -0.3810
0.260     -1.467      0.142      -0.890       0.128
B                                                               0.3840
0.713      0.539      0.590      -1.014       1.782
C                                                              -0.0901
0.672     -0.134      0.893      -1.406       1.226
```

```
D                                                                    0.8875
0.751      1.181      0.238      -0.585       2.360
E                                                                    1.0669
0.752      1.418      0.156      -0.407       2.541
F                                                                    0.2211
1.102      0.201      0.841      -1.939       2.381
G                                                                   -1.3949
1.255     -1.112      0.266      -3.854       1.064
M                                                                   -0.4634
0.655     -0.707      0.479      -1.747       0.821
male                                                                 0.2801
0.548      0.511      0.610      -0.795       1.355
Mr                                                                  -3.2986
0.593     -5.566      0.000      -4.460      -2.137
Others                                                              -2.9690
0.775     -3.829      0.000      -4.489      -1.449
Big                                                                 -3.1760
1.100     -2.889      0.004      -5.331      -1.021
Small                                                               -0.4798
0.397     -1.209      0.226      -1.257       0.298
==============================================================================
==================================================
"""
```

```
X = df.drop(['Survived','Miss_Mrs_Ms', 'Fare'], axis = 1)

predictions = m1.predict(X)
thresholds = np.arange(0, 100)/100
best_thres = 0
best_score = 0
for thresh in thresholds:
    oofs_rounded = (predictions > thresh) * 1
    thresh_score = f1_score(df["Survived"], oofs_rounded)
    if thresh_score > best_score:
        best_score = thresh_score
        best_thres = thresh
print(f'Threshold {best_thres}: {best_score}')
```

```
Threshold 0.41: 0.7885714285714286
```

```
round_preds = (predictions > best_thres) * 1

print(confusion_matrix(df["Survived"], round_preds))
```

```
[[467  82]
 [ 66 276]]
```

Thus we have 492 True Negatives, 57 False Positives, 81 False Negatives, 261 True Positives

```
X.dtypes
```

```
Pclass                  int64
SibSp                   int64
Parch                   int64
Ticket_Frequency        int64
IsMarried               int64
Child                   int64
Age_band                int64
Fare_Range           category
Q                       uint8
S                       uint8
B                       uint8
C                       uint8
D                       uint8
E                       uint8
F                       uint8
G                       uint8
M                       uint8
male                    uint8
Mr                      uint8
Others                  uint8
Big                     uint8
Small                   uint8
dtype: object
```

Updates

```
# Create new feature Age_band*Pclass

df['Age_Class'] = df['Pclass']*df['Age_band']
```

Upsampling & Downsampling

```
# Using resample to upsample and downsample to the minority class

from sklearn.utils import resample

X_train = df.drop(['Survived','Miss_Mrs_Ms', 'Fare'], axis = 1)
y_train = df['Survived']
# Upsample minority class
X_train_u, y_train_u = resample(X_train[y_train == 1],
                                y_train[y_train == 1],
                                replace=True,
                                n_samples=X_train[y_train ==
0].shape[0],
                                random_state=1)

X_train_u = np.concatenate((X_train[y_train == 0], X_train_u))
y_train_u = np.concatenate((y_train[y_train == 0], y_train_u))
```

```
# Downsample majority class
X_train_d, y_train_d = resample(X_train[y_train == 0],
                                y_train[y_train == 0],
                                replace=True,
                                n_samples=X_train[y_train ==
1].shape[0],
                                random_state=1)
X_train_d = np.concatenate((X_train[y_train == 1], X_train_d))
y_train_d = np.concatenate((y_train[y_train == 1], y_train_d))


print("Original shape:", X_train.shape, y_train.shape)
print("Upsampled shape:", X_train_u.shape, y_train_u.shape)
#print ("SMOTE sample shape:", x_train_sm.shape, y_train_sm.shape)
print("Downsampled shape:", X_train_d.shape, y_train_d.shape)
```

```
Original shape: (891, 23) (891,)
Upsampled shape: (1098, 23) (1098,)
Downsampled shape: (684, 23) (684,)
```

```
# Creating the upsampled and downsampled dataframes

X_train_u = pd.DataFrame(X_train_u, columns=X.columns)
X_train_d = pd.DataFrame(X_train_d, columns=X.columns,
dtype=X.dtypes.values)
X_train_u['Survived'] = y_train_u
X_train_d['Survived'] = y_train_d
X_train_u['Fare_Range'] = X_train_u['Fare_Range'].astype('category')
X_train_d['Fare_Range'] = X_train_d['Fare_Range'].astype('category')
```

```
# Fitting on the upsampled data
m1 = smf.logit(
  formula='Survived ~ Pclass + SibSp + Parch + Ticket_Frequency +
IsMarried + Child + Age_band + Fare_Range + Q + S + B + C + D + E + F
+ G + M + male + Mr + Others + Big + Small + Age_Class',
  data=X_train_u) \
.fit()

m1.summary()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.381698
         Iterations: 35
```

```
C:\Users\hp\Anaconda3\lib\site-packages\statsmodels\base\model.py:606:
ConvergenceWarning: Maximum Likelihood optimization failed to
converge. Check mle_retvals
  ConvergenceWarning)
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

```
                           Logit Regression Results
==============================================================================
========
Dep. Variable:               Survived   No. Observations:
1098
Model:                          Logit   Df Residuals:
1047
Method:                           MLE   Df Model:
50
Date:                Thu, 02 Dec 2021   Pseudo R-squ.:
0.4493
Time:                        03:25:23   Log-Likelihood:
-419.10
converged:                       False   LL-Null:
-761.08
Covariance Type:            nonrobust   LLR p-value:
3.453e-112
==============================================================================
===================================================
                                                        coef
std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
---------------------------------------------------
Intercept                                                1.0938
1.500      0.729      0.466      -1.845       4.033
Pclass[T.2]                                              1.2351
1.469      0.841      0.400      -1.644       4.114
Pclass[T.3]                                             -0.5783
1.267     -0.456      0.648      -3.062       1.905
SibSp[T.1]                                              -0.0156
0.478     -0.033      0.974      -0.953       0.922
SibSp[T.2]                                               1.3456
0.720      1.868      0.062      -0.066       2.758
SibSp[T.3]                                               0.7408
1.232      0.601      0.548      -1.674       3.155
SibSp[T.4]                                               1.2711
1.297      0.980      0.327      -1.272       3.814
SibSp[T.5]                                               0.9788
7766.329      0.000      1.000   -1.52e+04    1.52e+04
SibSp[T.8]                                             -97.4643
1.04e+21  -9.39e-20      1.000   -2.04e+21    2.04e+21
Parch[T.1]                                              -0.5896
0.428     -1.379      0.168      -1.428       0.249
Parch[T.2]                                              -0.2703
0.590     -0.458      0.647      -1.427       0.887
Parch[T.3]                                               2.6476
1.649      1.606      0.108      -0.584       5.879
Parch[T.4]                                             -16.8059
```

```
1.17e+04     -0.001      0.999    -2.3e+04     2.3e+04
Parch[T.5]                                          -20.1932
3.67e+04     -0.001      1.000   -7.19e+04    7.18e+04
Parch[T.6]                                           1.0223
2.4e+04   4.27e-05      1.000    -4.7e+04     4.7e+04
Ticket_Frequency[T.2]                               -0.0660
0.301     -0.219      0.826      -0.656       0.524
Ticket_Frequency[T.3]                                0.2667
0.435      0.613      0.540      -0.587       1.120
Ticket_Frequency[T.4]                                0.0999
0.590      0.169      0.866      -1.056       1.256
Ticket_Frequency[T.5]                               -23.8209
4.38e+04     -0.001      1.000   -8.58e+04    8.57e+04
Ticket_Frequency[T.6]                               -18.5631
4831.678     -0.004      0.997   -9488.477    9451.351
Ticket_Frequency[T.7]                                1.7367
0.826      2.103      0.035       0.118       3.355
IsMarried[T.1]                                       0.6891
0.378      1.822      0.068      -0.052       1.430
Child[T.1]                                           1.8920
0.686      2.760      0.006       0.548       3.236
Age_band[T.1]                                        1.4812
nan        nan        nan         nan         nan
Age_band[T.2]                                        3.1079
nan        nan        nan         nan         nan
Age_band[T.3]                                        2.2155
nan        nan        nan         nan         nan
Age_band[T.4]                                       -16.7861
nan        nan        nan         nan         nan
Fare_Range[T.Interval(7.91, 14.454, closed='right')]      0.0299
0.293      0.102      0.919      -0.545       0.605
Fare_Range[T.Interval(14.454, 31.0, closed='right')]      0.0469
0.368      0.127      0.899      -0.675       0.769
Fare_Range[T.Interval(31.0, 512.329, closed='right')]    -0.0806
0.556     -0.145      0.885      -1.169       1.008
Q[T.1]                                               0.2060
0.390      0.529      0.597      -0.558       0.970
S[T.1]                                              -0.4811
0.246     -1.957      0.050      -0.963       0.001
B[T.1]                                               0.3569
0.686      0.520      0.603      -0.988       1.701
C[T.1]                                              -0.2542
0.618     -0.411      0.681      -1.465       0.957
D[T.1]                                               0.9620
0.702      1.370      0.171      -0.414       2.338
E[T.1]                                               1.0264
0.708      1.450      0.147      -0.361       2.414
F[T.1]                                              -0.5310
1.126     -0.472      0.637      -2.737       1.675
```

```
G[T.1]                                                    -0.8528
1.172     -0.728      0.467      -3.149       1.444
M[T.1]                                                    -0.6323
0.623     -1.016      0.310      -1.853       0.588
male[T.1]                                                  1.7295
0.652      2.652      0.008       0.451       3.008
Mr[T.1]                                                   -4.7763
0.689     -6.931      0.000      -6.127      -3.426
Others[T.1]                                               -4.5419
0.835     -5.440      0.000      -6.178      -2.906
Big[T.1]                                                  -3.6727
1.331     -2.759      0.006      -6.281      -1.064
Small[T.1]                                                -0.2821
0.571     -0.494      0.621      -1.401       0.837
Age_Class[T.1]                                             1.1688
nan        nan        nan         nan         nan
Age_Class[T.2]                                            -0.8542
nan        nan        nan         nan         nan
Age_Class[T.3]                                            -0.0887
nan        nan        nan         nan         nan
Age_Class[T.4]                                            -2.5753
nan        nan        nan         nan         nan
Age_Class[T.6]                                            -2.0160
nan        nan        nan         nan         nan
Age_Class[T.8]                                            -2.5628
nan        nan        nan         nan         nan
Age_Class[T.9]                                            -1.6949
nan        nan        nan         nan         nan
Age_Class[T.12]                                           -1.3583
nan        nan        nan         nan         nan
==========================================================================
====================================================
"""

# Fitting on the downsampled data
m1 = smf.logit(
  formula='Survived ~ Pclass + SibSp + Parch + Ticket_Frequency +
IsMarried + Child + Age_band + Fare_Range + Q + S + B + C + D + E + F
+ G + M + male + Mr + Others + Big + Small + Age_Class',
  data=X_train_d) \
.fit()

m1.summary()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.382638
         Iterations: 35

C:\Users\hp\Anaconda3\lib\site-packages\statsmodels\base\model.py:606:
ConvergenceWarning: Maximum Likelihood optimization failed to
```

```
converge. Check mle_retvals
  ConvergenceWarning)

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

==============================================================================
========
Dep. Variable:               Survived   No. Observations:
684
Model:                          Logit   Df Residuals:
635
Method:                           MLE   Df Model:
48
Date:                Thu, 02 Dec 2021   Pseudo R-squ.:
0.4480
Time:                        03:25:25   Log-Likelihood:
-261.72
converged:                       False   LL-Null:
-474.11
Covariance Type:            nonrobust   LLR p-value:
8.355e-62
==============================================================================
====================================================
                                                          coef
std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
----------------------------------------------------
Intercept                                                29.1405
7.77e+05   3.75e-05      1.000   -1.52e+06    1.52e+06
Pclass[T.2]                                             -25.3053
7.77e+05  -3.26e-05      1.000   -1.52e+06    1.52e+06
Pclass[T.3]                                             -26.9066
7.77e+05  -3.46e-05      1.000   -1.52e+06    1.52e+06
SibSp[T.1]                                               0.4606
0.624      0.738      0.461      -0.763       1.684
SibSp[T.2]                                               1.3198
0.950      1.389      0.165      -0.543       3.182
SibSp[T.3]                                               0.7750
1.727      0.449      0.654      -2.609       4.159
SibSp[T.4]                                               0.8682
1.870      0.464      0.642      -2.797       4.534
SibSp[T.5]                                              -1.8573
1.35e+04     -0.000      1.000   -2.65e+04    2.65e+04
Parch[T.1]                                              -0.2422
0.560     -0.432      0.665      -1.340       0.856
Parch[T.2]                                              -0.4156
0.741     -0.561      0.575      -1.869       1.037
Parch[T.3]                                               0.6588
```

```
1.923      0.343      0.732      -3.110       4.428
Parch[T.4]                                                  -0.4481
8728.678  -5.13e-05      1.000   -1.71e+04    1.71e+04
Parch[T.5]                                                   0.2163
2.198      0.098      0.922      -4.092       4.524
Ticket_Frequency[T.2]                                        0.0590
0.402      0.147      0.883      -0.730       0.847
Ticket_Frequency[T.3]                                        0.2407
0.514      0.468      0.640      -0.767       1.248
Ticket_Frequency[T.4]                                        1.4789
0.821      1.801      0.072      -0.131       3.088
Ticket_Frequency[T.5]                                      -26.6842
4.25e+05  -6.27e-05      1.000   -8.34e+05    8.34e+05
Ticket_Frequency[T.6]                                      -15.0687
1930.224     -0.008      0.994   -3798.238    3768.101
Ticket_Frequency[T.7]                                        1.9985
0.984      2.031      0.042       0.070       3.927
IsMarried[T.1]                                               1.1873
0.537      2.212      0.027       0.135       2.239
Child[T.1]                                                   0.1094
1.137      0.096      0.923      -2.119       2.337
Age_band[T.1]                                               14.0224
nan        nan        nan         nan         nan
Age_band[T.2]                                              -11.4310
nan        nan        nan         nan         nan
Age_band[T.3]                                              -12.4506
nan        nan        nan         nan         nan
Age_band[T.4]                                              -36.3436
nan        nan        nan         nan         nan
Fare_Range[T.Interval(7.91, 14.454, closed='right')]         0.1495
0.366      0.409      0.683      -0.568       0.867
Fare_Range[T.Interval(14.454, 31.0, closed='right')]         0.1472
0.463      0.318      0.751      -0.760       1.055
Fare_Range[T.Interval(31.0, 512.329, closed='right')]       -0.0163
0.699     -0.023      0.981      -1.387       1.355
Q[T.1]                                                      -0.2188
0.503     -0.435      0.664      -1.205       0.767
S[T.1]                                                      -0.6832
0.324     -2.111      0.035      -1.317      -0.049
B[T.1]                                                       0.5338
0.822      0.650      0.516      -1.076       2.144
C[T.1]                                                       0.3623
0.762      0.475      0.635      -1.132       1.857
D[T.1]                                                       1.2846
0.878      1.462      0.144      -0.437       3.006
E[T.1]                                                       1.6220
0.880      1.843      0.065      -0.103       3.347
F[T.1]                                                      -0.6997
1.242     -0.564      0.573      -3.134       1.734
```

```
G[T.1]                                                          23.7863
2.4e+05    9.9e-05      1.000   -4.71e+05    4.71e+05
M[T.1]                                                          -0.5800
0.801     -0.724      0.469      -2.150       0.990
male[T.1]                                                        0.8889
0.791      1.124      0.261      -0.662       2.440
Mr[T.1]                                                         -3.7853
0.828     -4.570      0.000      -5.409      -2.162
Others[T.1]                                                     -3.4673
0.996     -3.481      0.001      -5.420      -1.515
Big[T.1]                                                        -4.3302
1.867     -2.320      0.020      -7.989      -0.671
Small[T.1]                                                      -0.5077
0.737     -0.689      0.491      -1.951       0.936
Age_Class[T.1]                                                 -39.0641
nan        nan        nan         nan         nan
Age_Class[T.2]                                                 -15.0088
nan        nan        nan         nan         nan
Age_Class[T.3]                                                 -14.2374
nan        nan        nan         nan         nan
Age_Class[T.4]                                                   9.8245
nan        nan        nan         nan         nan
Age_Class[T.6]                                                  10.4136
nan        nan        nan         nan         nan
Age_Class[T.8]                                                  -4.3162
5.2e+08  -8.29e-09      1.000   -1.02e+09    1.02e+09
Age_Class[T.9]                                                  11.0429
nan        nan        nan         nan         nan
Age_Class[T.12]                                                 -4.8573
1.84e+09  -2.64e-09      1.000    -3.6e+09     3.6e+09
==============================================================================
==================================================
"""
```

```python
# Importing the required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')

import seaborn as sns

# Reading the data
df = pd.read_csv('adult.csv')

df.head()
```

```
   Age          Workclass  fnlwgt   education  education-num  \
0   39          State-gov   77516   Bachelors             13
1   50   Self-emp-not-inc   83311   Bachelors             13
2   38            Private  215646     HS-grad              9
3   53            Private  234721        11th              7
4   28            Private  338409   Bachelors             13

         maritalstatus          occupation    relationship    race
sex  \
0        Never-married        Adm-clerical   Not-in-family   White
Male
1   Married-civ-spouse     Exec-managerial         Husband   White
Male
2             Divorced   Handlers-cleaners   Not-in-family   White
Male
3   Married-civ-spouse   Handlers-cleaners         Husband   Black
Male
4   Married-civ-spouse      Prof-specialty            Wife   Black
Female

   capital-gain  capital-loss  hourspw   nativecountry  target
0          2174             0       40   United-States   <=50K
1             0             0       13   United-States   <=50K
2             0             0       40   United-States   <=50K
3             0             0       40   United-States   <=50K
4             0             0       40            Cuba   <=50K
```

```python
#Checking for null values and data types
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   maritalstatus   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  hourspw         32561 non-null  int64
 13  nativecountry   32561 non-null  object
 14  target          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

We do not have any null values!

```
#description of numerical variales
df.describe().transpose()
```

```
                  count           mean            std      min
25%  \
Age             32561.0      38.581647      13.640433     17.0
28.0
fnlwgt          32561.0  189778.366512  105549.977697  12285.0
117827.0
education-num   32561.0      10.080679       2.572720      1.0
9.0
capital-gain    32561.0    1077.648844    7385.292085      0.0
0.0
capital-loss    32561.0      87.303830     402.960219      0.0
0.0
hourspw         32561.0      40.437456      12.347429      1.0
40.0

                     50%       75%        max
Age                 37.0      48.0       90.0
fnlwgt          178356.0  237051.0  1484705.0
education-num       10.0      12.0       16.0
capital-gain         0.0       0.0    99999.0
capital-loss         0.0       0.0     4356.0
hourspw             40.0      45.0       99.0
```

```
#Plotting target distribution
f,ax=plt.subplots(1,2,figsize=(18,8))
df['target'].value_counts().plot.pie(ax = ax[0],
explode=[0,0.1],autopct='%1.1f%%',shadow=True)
ax[0].set_title('target')
```

```
ax[0].set_ylabel('')
sns.countplot('target',data=df,ax=ax[1])
ax[1].set_title('target')
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
Text(0.5, 1.0, 'target')
```

Overall 24.1 % people have a higher income than 50K

EDA and Feature Generation

```
#Checking the Columns or features
df.columns
```

```
Index(['Age', 'Workclass', 'fnlwgt', 'education', 'education-num',
       'maritalstatus', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hourspw', 'nativecountry',
'target'],
      dtype='object')
```

```
# find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']
print(categorical)
```

```
['Workclass', 'education', 'maritalstatus', 'occupation',
'relationship', 'race', 'sex', 'nativecountry', 'target']
```

```
# Find numerical variables

numerical = [var for var in df.columns if df[var].dtype !='O']
print(numerical)
```

```
['Age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss',
'hourspw']
```

```
# We check for values with '?'

for i in categorical:
    for j in df[i].value_counts().index:
        if j == ' ?':
            print(i)
            break
```

```
Workclass
occupation
nativecountry
```

```
#Unique classes in Workclass
df['Workclass'].unique()
```

```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

```
# Number of data points per unique value
df['Workclass'].value_counts()
```

```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: Workclass, dtype: int64
```

We see that there are 1836 '?' values, that are corrupt and should be treated.

```
#Replacing '?' with NaN values
df['Workclass'].replace(' ?', np.NaN, inplace= True)

df['Workclass'].value_counts()
```

```
 Private              22696
 Self-emp-not-inc      2541
 Local-gov             2093
 State-gov             1298
 Self-emp-inc          1116
 Federal-gov            960
 Without-pay             14
 Never-worked             7
Name: Workclass, dtype: int64
```

Checking Occupation Variable

```
# Repeating the same process with Occupation
df['occupation'].unique()
```

```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Craft-repair',
       ' Transport-moving', ' Farming-fishing', ' Machine-op-inspct',
       ' Tech-support', ' ?', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

```
df['occupation'].value_counts()
```

```
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
 ?                    1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64
```

Again we see that there are 1843 '?' counts

```
df['occupation'].replace(' ?', np.NaN, inplace=True)
```

```
df['occupation'].value_counts()
```

```
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
```

```
 Sales                 3650
 Other-service         3295
 Machine-op-inspct     2002
 Transport-moving      1597
 Handlers-cleaners     1370
 Farming-fishing        994
 Tech-support           928
 Protective-serv        649
 Priv-house-serv        149
 Armed-Forces             9
Name: occupation, dtype: int64
```

Checking native country

```
# Repeating the same process with Native Country
df['nativecountry'].unique()
```

```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' ?', ' 
Mexico',
       ' South', ' Puerto-Rico', ' Honduras', ' England', ' Canada',
       ' Germany', ' Iran', ' Philippines', ' Italy', ' Poland',
       ' Columbia', ' Cambodia', ' Thailand', ' Ecuador', ' Laos',
       ' Taiwan', ' Haiti', ' Portugal', ' Dominican-Republic',
       ' El-Salvador', ' France', ' Guatemala', ' China', ' Japan',
       ' Yugoslavia', ' Peru', ' Outlying-US(Guam-USVI-etc)', ' 
Scotland',
       ' Trinadad&Tobago', ' Greece', ' Nicaragua', ' Vietnam', ' 
Hong',
       ' Ireland', ' Hungary', ' Holand-Netherlands'], dtype=object)
```

```
df['nativecountry'].value_counts()
```

```
 United-States                 29170
 Mexico                          643
 ?                               583
 Philippines                     198
 Germany                         137
 Canada                          121
 Puerto-Rico                     114
 El-Salvador                     106
 India                           100
 Cuba                             95
 England                          90
 Jamaica                          81
 South                            80
 China                            75
 Italy                            73
 Dominican-Republic               70
 Vietnam                          67
 Guatemala                        64
```

```
 Japan                              62
 Poland                             60
 Columbia                           59
 Taiwan                             51
 Haiti                              44
 Iran                               43
 Portugal                           37
 Nicaragua                          34
 Peru                               31
 France                             29
 Greece                             29
 Ecuador                            28
 Ireland                            24
 Hong                               20
 Cambodia                           19
 Trinadad&Tobago                    19
 Laos                               18
 Thailand                           18
 Yugoslavia                         16
 Outlying-US(Guam-USVI-etc)         14
 Honduras                           13
 Hungary                            13
 Scotland                           12
 Holand-Netherlands                  1
Name: nativecountry, dtype: int64
```

```
df['nativecountry'].replace(' ?', np.NaN, inplace=True)

for i in categorical:
    for j in df[i].value_counts().index:
        if j == ' ?':
            print(i)
            break
```

After careful observation, we donot observe any '?' in any other features

```
df[categorical].isnull().sum()
```

```
Workclass        1836
education           0
maritalstatus       0
occupation       1843
relationship        0
race                0
sex                 0
nativecountry     583
target              0
dtype: int64
```

EDA

```
# Creating countplot for categorical variables with hue as target
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='Workclass',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,1,figsize=(17,8))
sns.countplot(x='education',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='maritalstatus',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,1,figsize=(24,8))
sns.countplot(x='occupation',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='relationship',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='race',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='sex',data=df,hue='target', ax = ax)
```

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

```
# Creating histograms and boxplot for numerical variables
f,ax=plt.subplots(1,2,figsize=(16,5.5))
print(ax.shape)
ax[0].hist(df.Age, edgecolor="black")
```

```
ax[0].set_xlabel('Age')
ax[0].set_ylabel('count')
sns.boxplot(x='target', y='Age', data=df, ax=ax[1])
plt.show()
```

```
(2,)
```

```
df.describe()
```

```
                Age        fnlwgt  education-num  capital-gain
capital-loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000
32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844
87.303830
std       13.640433  1.055500e+05       2.572720   7385.292085
402.960219
min       17.000000  1.228500e+04       1.000000      0.000000
0.000000
25%       28.000000  1.178270e+05       9.000000      0.000000
0.000000
50%       37.000000  1.783560e+05      10.000000      0.000000
0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000
0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000
4356.000000

            hourspw
count  32561.000000
mean      40.437456
std       12.347429
min        1.000000
25%       40.000000
50%       40.000000
```

```
75%        45.000000
max        99.000000
```

```
# Creating a new variable isChild for age less than 24
df['ischild'] = 0
df['ischild'][df['Age'] < 24] = 1
```

```
C:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing
imports until
```

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='ischild',data=df,hue='target', ax = ax)
```

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

```
# Creating a new variable seniorcitizen for age more than 50
df['seniorcitizen'] = 0
df['seniorcitizen'][df['Age'] > 50] = 1
```

```
C:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing
imports until
```

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='seniorcitizen',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,2,figsize=(16,5.5))
print(ax.shape)
ax[0].hist(df.fnlwgt, edgecolor="black")
ax[0].set_xlabel('fnlwgt')
ax[0].set_ylabel('count')
sns.boxplot(x='target', y='fnlwgt', data=df, ax=ax[1])
plt.show()
```

```
(2,)
```

```
df['fnlwgt'].describe()
```

```
count    3.256100e+04
mean     1.897784e+05
std      1.055500e+05
min      1.228500e+04
25%      1.178270e+05
50%      1.783560e+05
75%      2.370510e+05
max      1.484705e+06
Name: fnlwgt, dtype: float64
```

```
f,ax=plt.subplots(1,2,figsize=(16,5.5))
print(ax.shape)
ax[0].hist(df['education-num'], edgecolor="black")
ax[0].set_xlabel('education-num')
ax[0].set_ylabel('count')
sns.boxplot(x='target', y='education-num', data=df, ax=ax[1])
plt.show()
```

```
(2,)
```

```
df['highly_edu'] = 0
df['highly_edu'][df['education-num'] > 10] = 1
```

```
C:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='highly_edu',data=df,hue='target', ax = ax)
```

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

```
f,ax=plt.subplots(1,2,figsize=(16,5.5))
print(ax.shape)
ax[0].hist(df['capital-gain'], edgecolor="black")
ax[0].set_xlabel('capital-gain')
ax[0].set_ylabel('count')
sns.boxplot(x='target', y='capital-gain', data=df, ax=ax[1])
plt.show()
```

```
(2,)
```

```
f,ax=plt.subplots(1,2,figsize=(16,5.5))
print(ax.shape)
ax[0].hist(df['capital-loss'], edgecolor="black")
ax[0].set_xlabel('capital-loss')
ax[0].set_ylabel('count')
sns.boxplot(x='target', y='capital-loss', data=df, ax=ax[1])
plt.show()
```

```
(2,)
```

```
f,ax=plt.subplots(1,2,figsize=(16,5.5))
print(ax.shape)
ax[0].hist(df['hourspw'], edgecolor="black")
ax[0].set_xlabel('hourspw')
ax[0].set_ylabel('count')
sns.boxplot(x='target', y='hourspw', data=df, ax=ax[1])
plt.show()
```

```
(2,)
```

```
df['work_more'] = 0
df['work_more'][df['hourspw'] >= 40] = 1
```

```
C:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

```
f,ax=plt.subplots(1,1,figsize=(15,8))
sns.countplot(x='work_more',data=df,hue='target', ax = ax)
```

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

Handling Missing values

```
# print percentage of missing values in the categorical variables in
training set

df[categorical].isnull().mean()
```

```
Workclass        0.056386
education        0.000000
maritalstatus    0.000000
occupation       0.056601
relationship     0.000000
race             0.000000
sex              0.000000
nativecountry    0.017905
target           0.000000
dtype: float64
```

```
# imputing the missing values with the modes of each feature
df['Workclass'].fillna(df['Workclass'].mode()[0], inplace = True)
df['occupation'].fillna(df['occupation'].mode()[0], inplace = True)
df['nativecountry'].fillna(df['nativecountry'].mode()[0], inplace =
True)

# print percentage of missing values in the categorical variables in
training set

df[categorical].isnull().mean()
```

```
Workclass        0.0
education        0.0
maritalstatus    0.0
occupation       0.0
relationship     0.0
race             0.0
sex              0.0
nativecountry    0.0
target           0.0
dtype: float64
```

Updates

```
df.head()
```

```
   Age          Workclass  fnlwgt   education  education-num  \
0   39          State-gov   77516   Bachelors             13
1   50   Self-emp-not-inc   83311   Bachelors             13
2   38            Private  215646     HS-grad              9
3   53            Private  234721        11th              7
```

```
4   28            Private  338409   Bachelors             13

         maritalstatus          occupation    relationship    race
sex  \
0        Never-married        Adm-clerical   Not-in-family   White
Male
1   Married-civ-spouse     Exec-managerial         Husband   White
Male
2             Divorced   Handlers-cleaners   Not-in-family   White
Male
3   Married-civ-spouse   Handlers-cleaners         Husband   Black
Male
4   Married-civ-spouse      Prof-specialty            Wife   Black
Female

   capital-gain  capital-loss  hourspw   nativecountry  target
ischild  \
0          2174             0       40   United-States   <=50K
0
1             0             0       13   United-States   <=50K
0
2             0             0       40   United-States   <=50K
0
3             0             0       40   United-States   <=50K
0
4             0             0       40            Cuba   <=50K
0

   seniorcitizen  highly_edu  work_more
0              0           1          1
1              0           1          0
2              0           0          1
3              1           0          1
4              0           1          1
```

```python
#data can be train or test
#var name is variable name: should be passed as strings within ('')
# bins is list of numeric values like [0,6,10,11]
# group names is list of groups you want to create in list form
def bin_var(data, var, bins, group_names):
    bin_value = bins
    group = group_names
    data[var+'Cat'] = pd.cut(df[var], bin_value, labels=group)

bin_var(df, 'education-num', [0,6,11,16], ['Low', 'Medium', 'High'])
```

Created a new variable grouping according to education numbers ar high, med or low

```
bin_var(df, 'hourspw', [0,35,40,60,100], ['Low', 'Medium',
'High','VeryHigh'])
```

Classifying the occupation into Highly Skilled and low Skilled

```
occu=pd.crosstab(df['occupation'],df['target'],
margins=True).reset_index()

occu
```

target	occupation	<=50K	>50K	All
0	Adm-clerical	3263	507	3770
1	Armed-Forces	8	1	9
2	Craft-repair	3170	929	4099
3	Exec-managerial	2098	1968	4066
4	Farming-fishing	879	115	994
5	Handlers-cleaners	1284	86	1370
6	Machine-op-inspct	1752	250	2002
7	Other-service	3158	137	3295
8	Priv-house-serv	148	1	149
9	Prof-specialty	3933	2050	5983
10	Protective-serv	438	211	649
11	Sales	2667	983	3650
12	Tech-support	645	283	928
13	Transport-moving	1277	320	1597
14	All	24720	7841	32561

```
#creating a function to categorize skill
import re
def occup(x):
    if re.search('managerial', x):
        return 'Highskill'
    elif re.search('specialty',x):
        return 'Highskill'
    else:
        return 'Lowskill'

# Creating the occupation category feature
df['Occupa_cat']=df.occupation.apply(lambda x: x.strip()).apply(lambda
x: occup(x))

df['Occupa_cat'].value_counts()
```

```
Lowskill     22512
Highskill    10049
Name: Occupa_cat, dtype: int64
```

Race has been binned into White and others

```
pd.crosstab(df['race'],df['target'], margins=True)
```

```
target                 <=50K   >50K    All
race
 Amer-Indian-Eskimo     275     36    311
 Asian-Pac-Islander     763    276   1039
 Black                 2737    387   3124
 Other                  246     25    271
 White                20699   7117  27816
All                   24720   7841  32561
```

```
# Creating race category feature
df['Race_cat']=df['race'].apply(lambda x: x.strip())
df['Race_cat']=df['Race_cat'].apply(lambda x: 'White' if x=='White'
else 'Other')
```

Encoding the categorical variables

```
# find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']
print(categorical)
```

```
['Workclass', 'education', 'maritalstatus', 'occupation',
'relationship', 'race', 'sex', 'nativecountry', 'target',
'Occupa_cat', 'Race_cat']
```

```
# Find numerical variables

numerical = [var for var in df.columns if df[var].dtype !='O']
print(numerical)
```

```
['Age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss',
'hourspw', 'ischild', 'seniorcitizen', 'highly_edu', 'work_more',
'education-numCat', 'hourspwCat']
```

```
categorical
```

```
['Workclass',
 'education',
 'maritalstatus',
 'occupation',
 'relationship',
 'race',
 'sex',
 'nativecountry',
 'target',
 'Occupa_cat',
 'Race_cat']
```

```
df.head()
```

```
   Age          Workclass  fnlwgt   education  education-num  \
0   39          State-gov   77516   Bachelors             13
1   50   Self-emp-not-inc   83311   Bachelors             13
2   38            Private  215646     HS-grad              9
3   53            Private  234721        11th              7
4   28            Private  338409   Bachelors             13

         maritalstatus          occupation    relationship    race
sex  \
0        Never-married        Adm-clerical   Not-in-family   White
Male
1   Married-civ-spouse     Exec-managerial         Husband   White
Male
2             Divorced   Handlers-cleaners   Not-in-family   White
Male
3   Married-civ-spouse   Handlers-cleaners         Husband   Black
Male
4   Married-civ-spouse      Prof-specialty            Wife   Black
Female

   ...   nativecountry  target  ischild seniorcitizen highly_edu
work_more  \
0  ...   United-States   <=50K        0             0          1
1
1  ...   United-States   <=50K        0             0          1
0
2  ...   United-States   <=50K        0             0          0
1
3  ...   United-States   <=50K        0             1          0
1
4  ...            Cuba   <=50K        0             0          1
1

   education-numCat  hourspwCat  Occupa_cat Race_cat
0              High      Medium    Lowskill    White
1              High         Low   Highskill    White
2            Medium      Medium    Lowskill    White
3            Medium      Medium    Lowskill    Other
4              High      Medium   Highskill    Other

[5 rows x 23 columns]
```

```python
# import category encoders

import category_encoders as ce
# encode remaining variables with one-hot encoding

encoder = ce.OneHotEncoder(cols=['Workclass',
 'education',
 'maritalstatus',
```

```
 'occupation',
 'relationship',
 'race',
 'sex',
 'nativecountry',
 'Occupa_cat',
 'Race_cat', 'education-numCat', "hourspwCat"])

df = encoder.fit_transform(df)
```

Feature Selection Using Variance Threshold

Variance Threshold is a univariate approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples. As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by The below approach removes variable which have more than 80% values are either 0 or 1

```
df.columns

Index(['Age', 'Workclass_1', 'Workclass_2', 'Workclass_3',
'Workclass_4',
       'Workclass_5', 'Workclass_6', 'Workclass_7', 'Workclass_8',
'fnlwgt',
       ...
       'education-numCat_2', 'education-numCat_3', 'hourspwCat_1',
       'hourspwCat_2', 'hourspwCat_3', 'hourspwCat_4', 'Occupa_cat_1',
       'Occupa_cat_2', 'Race_cat_1', 'Race_cat_2'],
      dtype='object', length=121)

# Using Variance Threshold for feature selection
from sklearn.feature_selection import VarianceThreshold
def variance_threshold_select(df, thresh=0.0, na_replacement=-999):
    df1 = df.copy(deep=True) # Make a deep copy of the dataframe
    selector = VarianceThreshold(thresh) # passing Threshold
    selector.fit(df1) # Fill NA values as VarianceThreshold cannot
deal with those
    df2 = df.loc[:,selector.get_support(indices=False)] # Get new
dataframe with columns deleted that have NA values
    return df2

# Setting a 80 percent threshold
df2=variance_threshold_select(df.drop('target', axis=1), thresh=.8* (1
- .8))

print(df2.columns)
```

```
Index(['Age', 'Workclass_3', 'fnlwgt', 'education_2', 'education_6',
       'education-num', 'maritalstatus_1', 'maritalstatus_2',
'relationship_1',
       'relationship_2', 'sex_1', 'sex_2', 'capital-gain', 'capital-
loss',
       'hourspw', 'highly_edu', 'work_more', 'education-numCat_2',
       'education-numCat_3', 'hourspwCat_1', 'hourspwCat_2',
'hourspwCat_3',
       'Occupa_cat_1', 'Occupa_cat_2'],
      dtype='object')
```

```
# Checking the final length of the selected features
len(df2.columns)
```

```
24
```

Modelling the Naive Bayes Classifier

```
df.head()
```

```
   Age  Workclass_1  Workclass_2  Workclass_3  Workclass_4
Workclass_5  \
0   39            1            0            0            0
0
1   50            0            1            0            0
0
2   38            0            0            1            0
0
3   53            0            0            1            0
0
4   28            0            0            1            0
0

   Workclass_6  Workclass_7  Workclass_8  fnlwgt  ...  education-
numCat_2  \
0            0            0            0   77516  ...
0
1            0            0            0   83311  ...
0
2            0            0            0  215646  ...
1
3            0            0            0  234721  ...
1
4            0            0            0  338409  ...
0

   education-numCat_3  hourspwCat_1  hourspwCat_2  hourspwCat_3
hourspwCat_4  \
0                   1             0             1             0
0
```

```
1                    1             1             0             0
0
2                    0             0             1             0
0
3                    0             0             1             0
0
4                    1             0             1             0
0

   Occupa_cat_1  Occupa_cat_2  Race_cat_1  Race_cat_2
0             1             0           1           0
1             0             1           1           0
2             1             0           1           0
3             1             0           0           1
4             0             1           0           1

[5 rows x 121 columns]
```

```python
# importing Gaussian Naive Bayes, Accuracy score and train test split
from sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

model = GaussianNB()

# Splitting data into test train, using a 0.3 split
X = df2
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# check the shape of X_train and X_test
X_train.shape, X_test.shape
```

```
((22792, 24), (9769, 24))
```

```python
# Scaling the training and test feature values
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Using Random Search method to find the best hyperparameters
from sklearn.model_selection import RandomizedSearchCV
gnb = GaussianNB()
param_grid = {
    'var_smoothing': np.logspace(0,-9, num=100)
}
```

```
CV_rfc = RandomizedSearchCV(estimator=gnb,
param_distributions=param_grid, cv= 5,random_state=1)
CV_rfc.fit(X_train, y_train)
print(CV_rfc.best_params_)
```

```
{'var_smoothing': 3.5111917342151273e-09}
```

```
#Defining the model with tuned hyperparameters
model = GaussianNB(var_smoothing=CV_rfc.best_params_['var_smoothing'])

# Fitting the model on to the train set
model.fit(X_train,y_train)
```

```
GaussianNB(var_smoothing=3.5111917342151273e-09)
```

```
predict_train = model.predict(X_train)

# Accuracy Score on train dataset
accuracy_train = accuracy_score(y_train,predict_train)
print('accuracy_score on train dataset : ', accuracy_train)

predict_test = model.predict(X_test)

# Accuracy Score on test dataset
accuracy_test = accuracy_score(y_test,predict_test)
print('accuracy_score on test dataset : ', accuracy_test)
```

```
accuracy_score on train dataset :  0.8292383292383292
accuracy_score on test dataset :  0.8334527587265841
```

We see that the accuracy values for test and train are close, and thus no overfitting!

Confusion Matrix

```
# Print the Confusion Matrix and slice it into four pieces

from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix

cm = confusion_matrix(y_test, predict_test)
print(cm)

plot_confusion_matrix(model, X_test, y_test)
```

```
[[6865  590]
 [1037 1277]]
```

```
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\utils\
deprecation.py:87: FutureWarning: Function plot_confusion_matrix is
deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and
will be removed in 1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
```

```
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x1e59e344948>
```

```
print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
True Positives(TP) =  6865

True Negatives(TN) =  1277

False Positives(FP) =  590

False Negatives(FN) =  1037
```

Classification Report

```
#Prinitng the classification report using the sklearn metrics library
from sklearn.metrics import classification_report

print(classification_report(y_test, predict_test))
```

```
              precision    recall  f1-score   support

       <=50K       0.87      0.92      0.89      7455
        >50K       0.68      0.55      0.61      2314

    accuracy                           0.83      9769
   macro avg       0.78      0.74      0.75      9769
weighted avg       0.82      0.83      0.83      9769
```

ROC - AUC

```
#Getting predicted probabilities
pred_proba = model.predict_proba(X_test)[:, 1]

# plot ROC Curve

from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, pred_proba, pos_label = '
>50K')

plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, linewidth=2)

plt.plot([0,1], [0,1], 'k--' )

plt.title('ROC curve for Gaussian Naive Bayes Classifier for
Predicting Salaries')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()
```

```
# compute ROC AUC

from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y_test, pred_proba)

print('ROC AUC : {:.4f}'.format(ROC_AUC))

ROC AUC : 0.8733
```

As our ROC AUC value is close to 1, we can say that our classifer model is working well !

k-Fold Cross Validation

```
# Applying 10-Fold Cross Validation

from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, X_train, y_train, cv = 10,
scoring='accuracy')

print('Cross-validation scores:{}'.format(scores))

print('Cross-validation mean accuracy:{}'.format(scores.mean()))

Cross-validation scores:[0.82807018 0.82675439 0.83369899 0.8174638
0.82974989 0.8372093
 0.82448442 0.825362   0.82229048 0.84422993]
Cross-validation mean accuracy:0.8289313372285475
```

We see that the mean accuracy is close to the original one, and also there is not much deviation from the average for all the folds, thus we can say our model is not much reliant on the data on which it is being trained.

Using different threshold values

```
#Transforming the test set in 0 and 1
y_testing = np.zeros(y_test.shape)
y_testing[y_test == ' >50K'] = 1

#Computing accuracy for different thresholds and finding the best one.
from sklearn.metrics import accuracy_score, f1_score

thresholds = np.arange(0, 100)/100
best_thres = 0
best_score = 0
for thresh in thresholds:
    oofs_rounded = (pred_proba > thresh) * 1
    thresh_score = accuracy_score(y_testing, oofs_rounded)
    if thresh_score > best_score:
        best_score = thresh_score
        best_thres = thresh
print(f'Threshold {best_thres}: {best_score}')
```

```
Threshold 0.74: 0.8361142389190296
```

Thus using 0.8 as thershold and predicting salary to be >50K for probability greater than 0.8 gives us the best accuracy score of 0.835

As you can see below the number of columns have been reduced to 24 because of the the variance threshold. The removed columns have the same value in 80% of the observations

```python
# Importing the required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')

import seaborn as sns

# Reading the data
df = pd.read_csv('car_evaluation.csv', names=['buying', 'maint',
'doors', 'persons', 'lug_boot', 'safety', 'target'])

# Visualizing the dataframe
df.head()
```

```
  buying  maint doors persons lug_boot safety target
0  vhigh  vhigh     2       2    small    low  unacc
1  vhigh  vhigh     2       2    small    med  unacc
2  vhigh  vhigh     2       2    small   high  unacc
3  vhigh  vhigh     2       2      med    low  unacc
4  vhigh  vhigh     2       2      med    med  unacc
```

```python
# Checking the shape of the data
df.shape
```

```
(1728, 7)
```

```python
#Checking for null values and data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   target    1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```python
# Null alues
df.isnull().sum()
```

```
buying      0
maint       0
```

```
doors       0
persons     0
lug_boot    0
safety      0
target      0
dtype: int64
```

We do not have any null values!

```
#description
df.describe().transpose()
```

```
         count unique    top  freq
buying    1728      4  vhigh   432
maint     1728      4  vhigh   432
doors     1728      4      2   432
persons   1728      3      2   576
lug_boot  1728      3  small   576
safety    1728      3    low   576
target    1728      4  unacc  1210
```

```
#Plotting target distribution
f,ax=plt.subplots(1,2,figsize=(18,8))
df['target'].value_counts().plot.pie(ax = ax[0], explode=[0,0.1, 0.2,
0.3],autopct='%1.1f%%',shadow=False, textprops={'fontsize': 14})
ax[0].set_title('target')
ax[0].set_ylabel('')
sns.countplot('target',data=df,ax=ax[1])
ax[1].set_title('target')
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
Text(0.5, 1.0, 'target')
```

Pre-processing

```
#Checking the Columns or features
df.columns
```

```
Index(['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety',
'target'], dtype='object')
```

```
# find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']
print(categorical)
```

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety',
'target']
```

```
# Find numerical variables

numerical = [var for var in df.columns if df[var].dtype !='O']
print(numerical)
```

```
[]
```

```
# Printing all the unique classes in the features.
for col in categorical:
    print(df[col].value_counts())
```

```
vhigh    432
high     432
med      432
low      432
```

```
Name: buying, dtype: int64
vhigh    432
high     432
med      432
low      432
Name: maint, dtype: int64
2        432
3        432
4        432
5more    432
Name: doors, dtype: int64
2       576
4       576
more    576
Name: persons, dtype: int64
small    576
med      576
big      576
Name: lug_boot, dtype: int64
low     576
med     576
high    576
Name: safety, dtype: int64
unacc    1210
acc       384
good       69
vgood      65
Name: target, dtype: int64
```

EDA

```
# Creating pairwise list
cols = df.drop('target', axis = 1).columns
paircols = []
for i in range(len(cols)-1):
    for j in range(i+1, len(cols)):
        paircols.append([cols[i], cols[j]])
```

Univariate Analysis

```
# Histplots for each feature
f,ax=plt.subplots(2,3,figsize=(16,8))
for i, col in enumerate(cols):
    sns.histplot(binwidth=0.5, x=col, hue="target", data=df,
stat="count", multiple="stack", palette="ch:0.25", ax=ax[i%2, i%3])
```

Multivariate analysis

```
# Stripplot for each feature
f,ax=plt.subplots(5,3,figsize=(15,25))
j = -1
for i, col in enumerate(paircols):
    if i%5 == 0: j = j+1
    sns.stripplot(x =col[0], y =col[1], data = df,jitter = True, hue
='target', dodge = True, ax=ax[i%5, j],)
    ax[i%5, j].legend(bbox_to_anchor=(0.7, 0.95), loc=2)
```

unacc
acc
vgood
good
maint
buying
doors
persons
lug_boot
safety
vhigh
high
med
low
small
big
more
2
3
4
5more

Modelling the Decision Tree models

As the dataset has categorical features and hence they needs to be encoded in the appropriate form. There are two main method of encodig:

1. One hot encoding
2. Label encoding

As we have categorial features that are ordinal in nature i.e that can be ranked (ordered) hence label encoding will solve our purpose.Had there been nominal features we could have preferred one hot encoding.

```
# Getting X and y
X = df.drop('target', axis = 1)
y = df['target']

# importing necessary package for encoding our categorial features
import category_encoders as ce

encoder = ce.OrdinalEncoder(cols=cols)
x= encoder.fit_transform(X)

x.head()
```

```
   buying  maint  doors  persons  lug_boot  safety
0       1      1      1        1         1       1
1       1      1      1        1         1       2
2       1      1      1        1         1       3
3       1      1      1        1         2       1
4       1      1      1        1         2       2
```

```
# importing necessary packages
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
import scikitplot.metrics as skplt
from sklearn import tree
from sklearn.model_selection import train_test_split

# Splitting data into test train, using a 0.3 split
X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.3, random_state=42)

# Exploring class distribution under train ,crossvalidation dataset
print('Training Dataset',X_train.shape,y_train.shape)
print('\n Class label distribution in Training Set\
n',y_train.value_counts())
print('\n***********')
print("\n CrossValidation Dataset",X_test.shape,y_test.shape)
print('\nClass label distribution in Cross Validation Set\
```

```
n',y_test.value_counts())
print('\n***********')
```

```
Training Dataset (1209, 6) (1209,)

 Class label distribution in Training Set
 unacc    852
acc      266
good      50
vgood     41
Name: target, dtype: int64

***********

 CrossValidation Dataset (519, 6) (519,)

Class label distribution in Cross Validation Set
 unacc    358
acc      118
vgood     24
good      19
Name: target, dtype: int64

***********
```

```
# Using Grid search to find the best decision tree classifier with the
given set of parameters.
from sklearn.model_selection import GridSearchCV

parameters={'max_depth': list(range(1,10)),
            'min_samples_leaf' : list(range(2,10,1)),
            'min_samples_split': list(range(5,10,10)),
            }
model=GridSearchCV(DecisionTreeClassifier(class_weight='balanced'),par
ameters,n_jobs=-1,cv=10,scoring='accuracy', )
model.fit(X_train,y_train)
print('The best model is ', model.best_estimator_)
print("\n The best model parameters are ",model.best_params_)
print("\n The model accuracy on train set
is",model.score(X_train,y_train))
print("\n The model accuracy on test set
is",model.score(X_test,y_test))

y_predict=model.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
The best model is  DecisionTreeClassifier(class_weight='balanced',
max_depth=9, min_samples_leaf=2,
```

```
                        min_samples_split=5)

 The best model parameters are  {'max_depth': 9, 'min_samples_leaf':
2, 'min_samples_split': 5}

 The model accuracy on train set is 0.9834574028122415

 The model accuracy on test set is 0.9691714836223507


 Classification Report
              precision    recall  f1-score   support

         acc       0.98      0.90      0.94       118
        good       0.65      0.89      0.76        19
       unacc       1.00      0.99      1.00       358
       vgood       0.86      1.00      0.92        24

    accuracy                           0.97       519
   macro avg       0.87      0.95      0.90       519
weighted avg       0.97      0.97      0.97       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```
# Visualising Decision Tree
clf =
```

```
tree.DecisionTreeClassifier(class_weight='balanced',max_depth=9,min_sa
mples_leaf=2,min_samples_split=5)
clf.fit(X_train, y_train)
trgt=[' acc','unacc ','good',' vgood']

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (20,20),
dpi=300)
tree.plot_tree(clf,feature_names = cols, class_names=trgt,filled =
True, fontsize=25);
```

```
# Visualize the trained Decision Tree by export_graphviz() method

from sklearn.tree import export_graphviz
from sklearn import tree
from IPython.display import SVG
```

```
from graphviz import Source
from IPython.display import display

graph = Source(tree.export_graphviz(clf ,feature_names = cols,
class_names = trgt, max_depth = 9, filled = True, ))
display(SVG(graph.pipe(format='svg')))
```

Random Forest Classifier

```
# Importing and defining the random search feature space.
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in range(200,2000,200)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(2, 30, num = 1)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [ 2,3, 4, 5, 7,10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4, 5]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800],
'max_features': ['auto', 'sqrt'], 'max_depth': [2, None],
'min_samples_split': [2, 3, 4, 5, 7, 10], 'min_samples_leaf': [1, 2,
4, 5], 'bootstrap': [True, False]}
```

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
```

```
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available
cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions =
random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs
= -1)
# Fit the random search model
rf_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(),
n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [2, None],
                                        'max_features': ['auto',
'sqrt'],
                                        'min_samples_leaf': [1, 2, 4,
5],
                                        'min_samples_split': [2, 3, 4,
5, 7,
                                                              10],
                                        'n_estimators': [200, 400,
600, 800,
                                                         1000, 1200,
1400, 1600,
                                                         1800]},
                   random_state=42, verbose=2)
```

```
# Printing the best hyperparameters

rf_random.best_params_
```

```
{'n_estimators': 800,
 'min_samples_split': 3,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': None,
 'bootstrap': False}
```

```
# Checking accuracies on the train and test data, predicting on the
test data.

print('The best model is ', rf_random.best_estimator_)
print("\n The best model parameters are ",rf_random.best_params_)
print("\n The model accuracy on train set
is",rf_random.score(X_train,y_train))
print("\n The model accuracy on test set
is",rf_random.score(X_test,y_test))
```

```
y_predict=rf_random.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
The best model is  RandomForestClassifier(bootstrap=False,
min_samples_split=3, n_estimators=800)

 The best model parameters are  {'n_estimators': 800,
'min_samples_split': 3, 'min_samples_leaf': 1, 'max_features': 'auto',
'max_depth': None, 'bootstrap': False}

 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9653179190751445


 Classification Report
              precision    recall  f1-score   support

         acc       0.95      0.91      0.93       118
        good       0.74      0.89      0.81        19
       unacc       0.99      0.99      0.99       358
       vgood       0.88      0.88      0.88        24

    accuracy                           0.97       519
   macro avg       0.89      0.92      0.90       519
weighted avg       0.97      0.97      0.97       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```
# Defining the model with the best parameters
clf = RandomForestClassifier(bootstrap=True, max_depth=None,
max_features='auto', min_samples_leaf=1, min_samples_split=3,
n_estimators = 1000)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier(min_samples_split=3, n_estimators=1000)
```

```
print("\n The model accuracy on train set
is",clf.score(X_train,y_train))
print("\n The model accuracy on test set is",clf.score(X_test,y_test))

y_predict=clf.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9653179190751445


 Classification Report
              precision    recall  f1-score   support

         acc       0.94      0.92      0.93       118
        good       0.74      0.89      0.81        19
```

```
       unacc       0.99      0.99      0.99       358
       vgood       0.91      0.88      0.89        24

    accuracy                           0.97       519
   macro avg       0.90      0.92      0.91       519
weighted avg       0.97      0.97      0.97       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```
# view the feature scores

feature_scores = pd.Series(clf.feature_importances_,
index=X_train.columns).sort_values(ascending=False)

feature_scores
```

```
safety      0.303113
persons     0.244645
maint       0.155830
buying      0.150934
lug_boot    0.086554
doors       0.058924
dtype: float64
```

```
# Creating a seaborn bar plot
```

```
sns.barplot(x=feature_scores, y=feature_scores.index)
# Add labels to the graph

plt.xlabel('Feature Importance Score')
plt.ylabel('Features')

# Add title to the graph
plt.title("Visualizing Important Features")

# Visualize the graph
plt.show()
```

Building a model on selected features without the least important feature

```
# declare feature vector and target variable, without the least
important feature

X1 = df.drop(['target', 'doors'], axis=1)

y1 = df['target']

# split data into training and testing sets

from sklearn.model_selection import train_test_split

X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1,
test_size = 0.33, random_state = 42)
```

```
# encode categorical variables with ordinal encoding

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'persons',
'lug_boot', 'safety'])


X_train1 = encoder.fit_transform(X_train1)

X_test1 = encoder.transform(X_test1)

# modelling without the least important feature
clf = RandomForestClassifier(bootstrap=True, max_depth=None,
max_features='auto', min_samples_leaf=1, min_samples_split=3,
n_estimators = 1000)
clf.fit(X_train1, y_train1)
```

```
RandomForestClassifier(min_samples_split=3, n_estimators=1000)
```

```
print("\n The model accuracy on train set
is",clf.score(X_train1,y_train1))
print("\n The model accuracy on test set
is",clf.score(X_test1,y_test1))

y_predict1=clf.predict(X_test1)
accuracy=accuracy_score(y_test1,y_predict1,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test1,y_predict1))
skplt.plot_confusion_matrix(y_test1,y_predict1)
```

```
 The model accuracy on train set is 0.9688850475367329

 The model accuracy on test set is 0.9334500875656743


 Classification Report
              precision    recall  f1-score   support

         acc       0.89      0.84      0.86       129
        good       0.56      0.90      0.69        20
       unacc       0.98      0.97      0.98       397
       vgood       0.80      0.80      0.80        25

    accuracy                           0.93       571
   macro avg       0.81      0.88      0.83       571
weighted avg       0.94      0.93      0.94       571


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

Xgboost

```
# import machine learning libraries
import xgboost as xgb

# import packages for hyperparameters tuning
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe

# Defining the feature space for search
space={'max_depth': hp.quniform("max_depth", 3, 18, 1),
        'gamma': hp.uniform ('gamma', 1,9),
        'reg_alpha' : hp.quniform('reg_alpha', 40,180,1),
        'reg_lambda' : hp.uniform('reg_lambda', 0,1),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0.5,1),
        'min_child_weight' : hp.quniform('min_child_weight', 0, 10,
1),
        'n_estimators': 180,
        'seed': 42
    }

# Deifning the objective function to minimize as this is required by
hyperopt library
def objective(space):
    clf=xgb.XGBClassifier(
                    n_estimators =space['n_estimators'], max_depth =
int(space['max_depth']), gamma = space['gamma'],
                    reg_alpha =
int(space['reg_alpha']),min_child_weight=int(space['min_child_weight']
```

```
),
                     colsample_bytree=int(space['colsample_bytree']))

    evaluation = [( X_train, y_train), ( X_test, y_test)]

    clf.fit(X_train, y_train,
            eval_set=evaluation, eval_metric = 'merror',
            early_stopping_rounds=10,verbose=2)


    pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, pred)
    print ("SCORE:", accuracy)
    return {'loss': -accuracy, 'status': STATUS_OK }

# Getting the optimum
trials = Trials()

best_hyperparams = fmin(fn = objective,
                        space = space,
                        algo = tpe.suggest,
                        max_evals = 100,
                        trials = trials)
```

```
   0%|
| 0/100 [00:00<?, ?trial/s, best loss=?]
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

[4]   validation_0-merror:0.29529validation_1-merror:0.31021

[6]   validation_0-merror:0.29529validation_1-merror:0.31021

[8]   validation_0-merror:0.29529validation_1-merror:0.31021

[9]   validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

  2%|▉                                                  | 2/100
[00:00<00:35,  2.76trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

  4%|▉                                                  | 4/100
[00:01<00:19,  4.86trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

  6%|██                                                  | 6/100
[00:01<00:15,  5.92trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

  8%|████                                                | 8/100
[00:01<00:14,  6.42trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 11%|▉                                                  | 11/100
[00:01<00:10,  8.27trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
```

```
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

 12%|└└└
```

```
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

 14%|██████                                                | 14/100
[00:02<00:09,  8.95trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 15%|┃┃┃┃┃┃┃                                                | 15/100
[00:03<00:28,  3.01trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

 16%|┃┃┃┃┃┃┃                                                | 16/100
[00:03<00:29,  2.84trial/s, best loss: -0.6897880539499036]
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 18%|┃┃┃┃┃┃┃┃                                          | 18/100
[00:03<00:21,  3.78trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 20%|██████████                                          | 20/100
[00:04<00:19,  4.16trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 22%|██████████                                        | 22/100
[00:04<00:18,  4.17trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[10]  validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

 24%|╟╟╟╟╟╟╟╟╟╟╟╟                                        | 24/100
[00:05<00:15,  5.02trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

[4]   validation_0-merror:0.29529validation_1-merror:0.31021

[6]   validation_0-merror:0.29529validation_1-merror:0.31021

[8]   validation_0-merror:0.29529validation_1-merror:0.31021

[9]   validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

 26%|╟╟╟╟╟╟╟╟╟╟╟╟                                        | 26/100
[00:05<00:15,  4.71trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

[4]   validation_0-merror:0.29529validation_1-merror:0.31021
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 27%|╫╫╫╫╫╫╫╫╫╫╫╫                                        | 27/100
[00:05<00:14,  4.95trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 29%|▉▉▉▉▉▉▉▉▉▉▉▉▉                                      | 29/100
[00:06<00:13,  5.18trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

 30%|[redacted]| 30/100
[00:06<00:13,  5.28trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 31%|[redacted]| 31/100
[00:07<00:24,  2.83trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

 32%|└└└
```

```
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 35%|╫╫╫
```

```
[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 39%|╫╫╫
```

```
[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 41%|┃┃┃┃┃┃┃┃┃┃┃┃┃┃┃┃┃┃┃┃                                | 41/100
[00:09<00:11,  5.32trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 45%|╫╫╫
```

```
[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 46%|██████████████████                                | 46/100
[00:10<00:10,  5.18trial/s, best loss: -0.6897880539499036]


C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
 48%|                                                  | 48/100
[00:11<00:20,  2.51trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

[4]   validation_0-merror:0.29529validation_1-merror:0.31021

[6]   validation_0-merror:0.29529validation_1-merror:0.31021

[8]   validation_0-merror:0.29529validation_1-merror:0.31021

[10]  validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

 48%|                                                  | 48/100
[00:11<00:20,  2.51trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


0.6897880539499036

[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

[4]   validation_0-merror:0.29529validation_1-merror:0.31021

[6]   validation_0-merror:0.29529validation_1-merror:0.31021

[8]   validation_0-merror:0.29529validation_1-merror:0.31021

[10]  validation_0-merror:0.29529validation_1-merror:0.31021
```

```
SCORE:

0.6897880539499036

 50%|                                                  | 50/100
[00:12<00:16,  3.12trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 51%|                                                  | 51/100
[00:12<00:13,  3.62trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 53%|╫╫╫
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 55%|██████████████████████                                  | 55/100
[00:13<00:09,  4.88trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

 57%|╫╫╫
```

```
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 59%|▇▇▇
```

```
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

 61%|██████████████████████████                    | 61/100
[00:14<00:06,  6.16trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
```

```
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 62%|██████████████████████████                  | 62/100
[00:14<00:06,  6.11trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[9]	validation_0-merror:0.29529	validation_1-merror:0.31021
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

 64%|██████████████████████████████                 | 64/100
[00:16<00:16,  2.22trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 66%|██████████████████████████                | 66/100
[00:16<00:11,  3.08trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 68%|██████████████████████████                | 68/100
[00:16<00:08,  3.94trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 70%|██████████████████████████████                | 70/100
[00:17<00:06,  4.75trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[6]	validation_0-merror:0.29529	validation_1-merror:0.31021
[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 74%|██████████████████████████████                | 74/100
[00:17<00:04,  5.97trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

 75%|╫╫╫
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 78%|┃┃┃
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 80%|┃┃┃
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

 82%|╫╫╫
```

```
[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

 84%|                                        | 84/100
[00:21<00:03,  5.27trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
```

```
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
```

```
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[10]	validation_0-merror:0.29529	validation_1-merror:0.31021
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

 88%|▉▉▉
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021
[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 92%|▉▉▉
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 94%|▉▉▉
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 96%|□□□
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 98%|▉▉▉
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

100%|██████████████████████████████████████████| 100/100
[00:24<00:00,  4.14trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```python
# Getting the best model
print("The best hyperparameters are : ","\n")
print(best_hyperparams)
```

```
The best hyperparameters are :

{'colsample_bytree': 0.8735300604984144, 'gamma': 4.042605283938188,
'max_depth': 15.0, 'min_child_weight': 3.0, 'reg_alpha': 95.0,
'reg_lambda': 0.4425426723169459}
```

```python
# Best model
clf=xgb.XGBClassifier(best_hyperparams)
```

```
evaluation = [( X_train, y_train), ( X_test, y_test)]
clf.fit(X_train, y_train,
            eval_set=evaluation,
            early_stopping_rounds=10,verbose=2)
```

```
[05:21:31] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.
[0]	validation_0-mlogloss:0.94153	validation_1-mlogloss:0.94379
[2]	validation_0-mlogloss:0.53773	validation_1-mlogloss:0.55620
[4]	validation_0-mlogloss:0.34786	validation_1-mlogloss:0.37244
[6]	validation_0-mlogloss:0.23741	validation_1-mlogloss:0.26699
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\core.py:502:
FutureWarning: Pass `objective` as keyword args.  Passing these as
positional arguments will be considered as error in future releases.
  format(", ".join(args_msg)), FutureWarning
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[8]	validation_0-mlogloss:0.17458	validation_1-mlogloss:0.21000
[10]	validation_0-mlogloss:0.13339	validation_1-mlogloss:0.17272
[12]	validation_0-mlogloss:0.10732	validation_1-mlogloss:0.14942
[14]	validation_0-mlogloss:0.08744	validation_1-mlogloss:0.12898
[16]	validation_0-mlogloss:0.07224	validation_1-mlogloss:0.11262
[18]	validation_0-mlogloss:0.06176	validation_1-mlogloss:0.10467
[20]	validation_0-mlogloss:0.05197	validation_1-mlogloss:0.09462
[22]	validation_0-mlogloss:0.04259	validation_1-mlogloss:0.08466
[24]	validation_0-mlogloss:0.03656	validation_1-mlogloss:0.08024
[26]	validation_0-mlogloss:0.03204	validation_1-mlogloss:0.07596
[28]	validation_0-mlogloss:0.02850	validation_1-mlogloss:0.07431
[30]	validation_0-mlogloss:0.02557	validation_1-mlogloss:0.07131
[32]	validation_0-mlogloss:0.02324	validation_1-mlogloss:0.07019
[34]	validation_0-mlogloss:0.02097	validation_1-mlogloss:0.06757
[36]	validation_0-mlogloss:0.01961	validation_1-mlogloss:0.06629
[38]	validation_0-mlogloss:0.01840	validation_1-mlogloss:0.06548
[40]	validation_0-mlogloss:0.01672	validation_1-mlogloss:0.06358
[42]	validation_0-mlogloss:0.01589	validation_1-mlogloss:0.06396
[44]	validation_0-mlogloss:0.01541	validation_1-mlogloss:0.06414
[46]	validation_0-mlogloss:0.01478	validation_1-mlogloss:0.06455
[48]	validation_0-mlogloss:0.01420	validation_1-mlogloss:0.06383
[50]	validation_0-mlogloss:0.01369	validation_1-mlogloss:0.06387
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1,
enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1,
missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, objective='multi:softprob',
predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1,
scale_pos_weight=None,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

```
print("\n The model accuracy on train set
is",clf.score(X_train,y_train))
print("\n The model accuracy on test set is",clf.score(X_test,y_test))

y_predict=clf.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9807321772639692


 Classification Report
              precision    recall  f1-score   support

         acc       0.97      0.95      0.96       118
        good       0.77      0.89      0.83        19
       unacc       1.00      1.00      1.00       358
       vgood       0.92      0.96      0.94        24

    accuracy                           0.98       519
   macro avg       0.92      0.95      0.93       519
weighted avg       0.98      0.98      0.98       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

Updates

```
import category_encoders as ce
encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors',
'persons', 'lug_boot', 'safety'])
df = encoder.fit_transform(df)

from sklearn.preprocessing import PolynomialFeatures

# summarize the shape of the dataset
print(df.shape)
# summarize each variable
display(df.head(2))

# perform a polynomial features transform of the dataset
trans = PolynomialFeatures(degree=4)
data = trans.fit_transform(df.iloc[:,:-1])

# convert the array back to a dataframe
dataset = pd.DataFrame(data)

# summarize
display(dataset.head(2))
```

```
(1728, 7)
```

```
   buying  maint  doors  persons  lug_boot  safety target
0       1      1      1        1         1       1  unacc
1       1      1      1        1         1       2  unacc
```

```
   0    1    2    3    4    5    6    7    8    9    ...  200  201
202  203  \
0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0
1.0  1.0
1  1.0  1.0  1.0  1.0  1.0  1.0  2.0  1.0  1.0  1.0  ...  4.0  1.0
2.0  4.0

   204  205  206  207  208   209
0  1.0  1.0  1.0  1.0  1.0   1.0
1  8.0  1.0  2.0  4.0  8.0  16.0

[2 rows x 210 columns]
```

```python
# Getting X and y
X = dataset
y = df['target']

X.head()
```

```
   0    1    2    3    4    5    6    7    8    9    ...  200  201
202  203  \
0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0
1.0  1.0
1  1.0  1.0  1.0  1.0  1.0  1.0  2.0  1.0  1.0  1.0  ...  4.0  1.0
2.0  4.0
2  1.0  1.0  1.0  1.0  1.0  1.0  3.0  1.0  1.0  1.0  ...  9.0  1.0
3.0  9.0
3  1.0  1.0  1.0  1.0  1.0  2.0  1.0  1.0  1.0  1.0  ...  1.0  8.0
4.0  2.0
4  1.0  1.0  1.0  1.0  1.0  2.0  2.0  1.0  1.0  1.0  ...  4.0  8.0
8.0  8.0

    204   205   206   207   208   209
0   1.0   1.0   1.0   1.0   1.0   1.0
1   8.0   1.0   2.0   4.0   8.0  16.0
2  27.0   1.0   3.0   9.0  27.0  81.0
3   1.0  16.0   8.0   4.0   2.0   1.0
4   8.0  16.0  16.0  16.0  16.0  16.0

[5 rows x 210 columns]
```

```python
# importing necessary packages
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
import scikitplot.metrics as skplt
from sklearn import tree
from sklearn.model_selection import train_test_split
```

```python
# Splitting data into test train, using a 0.3 split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Exploring class distribution under train ,crossvalidation dataset
print('Training Dataset',X_train.shape,y_train.shape)
print('\n Class label distribution in Training Set\
n',y_train.value_counts())
print('\n***********')
print("\n CrossValidation Dataset",X_test.shape,y_test.shape)
print('\nClass label distribution in Cross Validation Set\
n',y_test.value_counts())
print('\n***********')
```

```
Training Dataset (1209, 210) (1209,)

 Class label distribution in Training Set
 unacc    852
acc      266
good      50
vgood     41
Name: target, dtype: int64

***********

 CrossValidation Dataset (519, 210) (519,)

Class label distribution in Cross Validation Set
 unacc    358
acc      118
vgood     24
good      19
Name: target, dtype: int64

***********
```

```python
# Using Grid search to find the best decision tree classifier with the
given set of parameters.
from sklearn.model_selection import GridSearchCV

parameters={'max_depth': list(range(1,10)),
            'min_samples_leaf' : list(range(2,10,1)),
            'min_samples_split': list(range(5,10,10)),
            }
model=GridSearchCV(DecisionTreeClassifier(class_weight='balanced'),par
ameters,n_jobs=-1,cv=10,scoring='accuracy', )
model.fit(X_train,y_train)
print('The best model is ', model.best_estimator_)
print("\n The best model parameters are ",model.best_params_)
print("\n The model accuracy on train set
is",model.score(X_train,y_train))
```

```
print("\n The model accuracy on test set
is",model.score(X_test,y_test))

y_predict=model.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
The best model is  DecisionTreeClassifier(class_weight='balanced',
max_depth=9, min_samples_leaf=2,
                       min_samples_split=5)

 The best model parameters are  {'max_depth': 9, 'min_samples_leaf':
2, 'min_samples_split': 5}

 The model accuracy on train set is 0.9867659222497932

 The model accuracy on test set is 0.976878612716763


 Classification Report
              precision    recall  f1-score   support

         acc       0.96      0.93      0.95       118
        good       0.79      1.00      0.88        19
       unacc       0.99      0.99      0.99       358
       vgood       1.00      1.00      1.00        24

    accuracy                           0.98       519
   macro avg       0.94      0.98      0.96       519
weighted avg       0.98      0.98      0.98       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

Using SMOTE for Upsampling

```
X_train

      0    1    2    3    4    5    6     7     8     9    ...   200
201  \
1178  1.0  3.0  3.0  4.0  2.0  3.0  3.0   9.0   9.0  12.0  ...  36.0
54.0
585   1.0  2.0  2.0  2.0  3.0  1.0  1.0   4.0   4.0   4.0  ...   9.0
3.0
1552  1.0  4.0  3.0  2.0  2.0  2.0  2.0  16.0  12.0   8.0  ...  16.0
16.0
1169  1.0  3.0  3.0  4.0  1.0  3.0  3.0   9.0   9.0  12.0  ...   9.0
27.0
1033  1.0  3.0  2.0  3.0  1.0  3.0  2.0   9.0   6.0   9.0  ...   4.0
27.0
...   ...  ...  ...  ...  ...  ...  ...   ...   ...   ...  ...   ...
...
1130  1.0  3.0  3.0  2.0  3.0  2.0  3.0   9.0   9.0   6.0  ...  81.0
24.0
1294  1.0  3.0  4.0  4.0  3.0  3.0  2.0   9.0  12.0  12.0  ...  36.0
81.0
860   1.0  2.0  4.0  4.0  3.0  2.0  3.0   4.0   8.0   8.0  ...  81.0
24.0
1459  1.0  4.0  2.0  3.0  1.0  1.0  2.0  16.0   8.0  12.0  ...   4.0
1.0
1126  1.0  3.0  3.0  2.0  3.0  1.0  2.0   9.0   9.0   6.0  ...  36.0
3.0
```

```
       202   203   204   205   206   207   208   209
1178  54.0  54.0  54.0  81.0  81.0  81.0  81.0  81.0
585    3.0   3.0   3.0   1.0   1.0   1.0   1.0   1.0
1552  16.0  16.0  16.0  16.0  16.0  16.0  16.0  16.0
1169  27.0  27.0  27.0  81.0  81.0  81.0  81.0  81.0
1033  18.0  12.0   8.0  81.0  54.0  36.0  24.0  16.0
...    ...   ...   ...   ...   ...   ...   ...   ...
1130  36.0  54.0  81.0  16.0  24.0  36.0  54.0  81.0
1294  54.0  36.0  24.0  81.0  54.0  36.0  24.0  16.0
860   36.0  54.0  81.0  16.0  24.0  36.0  54.0  81.0
1459   2.0   4.0   8.0   1.0   2.0   4.0   8.0  16.0
1126   6.0  12.0  24.0   1.0   2.0   4.0   8.0  16.0

[1209 rows x 210 columns]
```

```
y_train
```

```
1178    vgood
585     unacc
1552      acc
1169    unacc
1033    unacc
        ...
1130    vgood
1294     good
860       acc
1459    unacc
1126      acc
Name: target, Length: 1209, dtype: object
```

```
from imblearn.over_sampling import SMOTE
from collections import Counter
from matplotlib import pyplot
from sklearn.preprocessing import LabelEncoder

# label encode the target variable
y = LabelEncoder().fit_transform(y_train)
# transform the dataset
oversample = SMOTE()
X, y = oversample.fit_resample(X_train, y)
# summarize distribution
counter = Counter(y)
for k,v in counter.items():
     per = v / len(y) * 100
     print('Class=%d, n=%d (%.3f%%)' % (k, v, per))
# plot the distribution
pyplot.bar(counter.keys(), counter.values())
pyplot.show()
```

```
Class=3, n=852 (25.000%)
Class=2, n=852 (25.000%)
Class=0, n=852 (25.000%)
Class=1, n=852 (25.000%)
```

```
X_train = X
y_train = y

y_test = LabelEncoder().fit_transform(y_test)

# Using Grid search to find the best decision tree classifier with the
given set of parameters.
from sklearn.model_selection import GridSearchCV

parameters={'max_depth': list(range(1,10)),
            'min_samples_leaf' : list(range(2,10,1)),
            'min_samples_split': list(range(5,10,10)),
            }
model=GridSearchCV(DecisionTreeClassifier(class_weight='balanced'),par
ameters,n_jobs=-1,cv=10,scoring='accuracy', )
model.fit(X_train,y_train)
print('The best model is ', model.best_estimator_)
print("\n The best model parameters are ",model.best_params_)
print("\n The model accuracy on train set
is",model.score(X_train,y_train))
print("\n The model accuracy on test set
is",model.score(X_test,y_test))

y_predict=model.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
```

```
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
 The model accuracy on test set is 0.9710982658959537


 Classification Report
              precision    recall  f1-score   support

           0       0.98      0.89      0.93       118
           1       0.66      1.00      0.79        19
           2       0.99      0.99      0.99       358
           3       1.00      1.00      1.00        24

    accuracy                           0.97       519
   macro avg       0.91      0.97      0.93       519
weighted avg       0.98      0.97      0.97       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```python
# Importing the required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')

import seaborn as sns

# Reading the data
df = pd.read_csv('car_evaluation.csv', names=['buying', 'maint',
'doors', 'persons', 'lug_boot', 'safety', 'target'])

# Visualizing the dataframe
df.head()
```

```
  buying  maint doors persons lug_boot safety target
0  vhigh  vhigh     2       2    small    low  unacc
1  vhigh  vhigh     2       2    small    med  unacc
2  vhigh  vhigh     2       2    small   high  unacc
3  vhigh  vhigh     2       2      med    low  unacc
4  vhigh  vhigh     2       2      med    med  unacc
```

```python
# Checking the shape of the data
df.shape
```

```
(1728, 7)
```

```python
#Checking for null values and data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   target    1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```python
# Null alues
df.isnull().sum()
```

```
buying      0
maint       0
```

```
doors       0
persons     0
lug_boot    0
safety      0
target      0
dtype: int64
```

We do not have any null values!

```
#description
df.describe().transpose()
```

```
         count unique    top  freq
buying    1728      4  vhigh   432
maint     1728      4  vhigh   432
doors     1728      4      2   432
persons   1728      3      2   576
lug_boot  1728      3  small   576
safety    1728      3    low   576
target    1728      4  unacc  1210
```

```
#Plotting target distribution
f,ax=plt.subplots(1,2,figsize=(18,8))
df['target'].value_counts().plot.pie(ax = ax[0], explode=[0,0.1, 0.2,
0.3],autopct='%1.1f%%',shadow=False, textprops={'fontsize': 14})
ax[0].set_title('target')
ax[0].set_ylabel('')
sns.countplot('target',data=df,ax=ax[1])
ax[1].set_title('target')
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
Text(0.5, 1.0, 'target')
```

Pre-processing

```
#Checking the Columns or features
df.columns
```

```
Index(['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety',
'target'], dtype='object')
```

```
# find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']
print(categorical)
```

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety',
'target']
```

```
# Find numerical variables

numerical = [var for var in df.columns if df[var].dtype !='O']
print(numerical)
```

```
[]
```

```
# Printing all the unique classes in the features.
for col in categorical:
    print(df[col].value_counts())
```

```
vhigh    432
high     432
med      432
low      432
```

```
Name: buying, dtype: int64
vhigh    432
high     432
med      432
low      432
Name: maint, dtype: int64
2        432
3        432
4        432
5more    432
Name: doors, dtype: int64
2       576
4       576
more    576
Name: persons, dtype: int64
small    576
med      576
big      576
Name: lug_boot, dtype: int64
low     576
med     576
high    576
Name: safety, dtype: int64
unacc    1210
acc       384
good       69
vgood      65
Name: target, dtype: int64
```

EDA

```
# Creating pairwise list
cols = df.drop('target', axis = 1).columns
paircols = []
for i in range(len(cols)-1):
    for j in range(i+1, len(cols)):
        paircols.append([cols[i], cols[j]])
```

Univariate Analysis

```
# Histplots for each feature
f,ax=plt.subplots(2,3,figsize=(16,8))
for i, col in enumerate(cols):
    sns.histplot(binwidth=0.5, x=col, hue="target", data=df,
stat="count", multiple="stack", palette="ch:0.25", ax=ax[i%2, i%3])
```

Multivariate analysis

```
# Stripplot for each feature
f,ax=plt.subplots(5,3,figsize=(15,25))
j = -1
for i, col in enumerate(paircols):
    if i%5 == 0: j = j+1
    sns.stripplot(x =col[0], y =col[1], data = df,jitter = True, hue
='target', dodge = True, ax=ax[i%5, j],)
    ax[i%5, j].legend(bbox_to_anchor=(0.7, 0.95), loc=2)
```

unacc
acc
vgood
good
maint
buying
doors
persons
lug_boot
safety
vhigh
high
med
low
small
big
more
2
3
4
5more

Modelling the Decision Tree models

As the dataset has categorical features and hence they needs to be encoded in the appropriate form. There are two main method of encodig:

1. One hot encoding
2. Label encoding

As we have categorial features that are ordinal in nature i.e that can be ranked (ordered) hence label encoding will solve our purpose.Had there been nominal features we could have preferred one hot encoding.

```
# Getting X and y
X = df.drop('target', axis = 1)
y = df['target']

# importing necessary package for encoding our categorial features
import category_encoders as ce

encoder = ce.OrdinalEncoder(cols=cols)
x= encoder.fit_transform(X)

x.head()
```

```
   buying  maint  doors  persons  lug_boot  safety
0       1      1      1        1         1       1
1       1      1      1        1         1       2
2       1      1      1        1         1       3
3       1      1      1        1         2       1
4       1      1      1        1         2       2
```

```
# importing necessary packages
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
import scikitplot.metrics as skplt
from sklearn import tree
from sklearn.model_selection import train_test_split

# Splitting data into test train, using a 0.3 split
X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.3, random_state=42)

# Exploring class distribution under train ,crossvalidation dataset
print('Training Dataset',X_train.shape,y_train.shape)
print('\n Class label distribution in Training Set\
n',y_train.value_counts())
print('\n***********')
print("\n CrossValidation Dataset",X_test.shape,y_test.shape)
print('\nClass label distribution in Cross Validation Set\
```

```
n',y_test.value_counts())
print('\n***********')
```

```
Training Dataset (1209, 6) (1209,)

 Class label distribution in Training Set
 unacc    852
acc      266
good      50
vgood     41
Name: target, dtype: int64

***********

 CrossValidation Dataset (519, 6) (519,)

Class label distribution in Cross Validation Set
 unacc    358
acc      118
vgood     24
good      19
Name: target, dtype: int64

***********
```

Random Forest Classifier

```
# Importing and defining the random search feature space.
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in range(400,1400,200)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(2, 15, num = 1)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [ 2,3, 4, 5, 7,10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4, 5]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
```

```
                'min_samples_leaf': min_samples_leaf,
                'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [400, 600, 800, 1000, 1200], 'max_features': ['auto',
'sqrt'], 'max_depth': [2, None], 'min_samples_split': [2, 3, 4, 5, 7,
10], 'min_samples_leaf': [1, 2, 4, 5], 'bootstrap': [True, False]}
```

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available
cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions =
random_grid, n_iter = 50, cv = 3, verbose=2, random_state=42, n_jobs =
-1)
# Fit the random search model
rf_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
```

```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(),
n_iter=50,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [2, None],
                                        'max_features': ['auto',
'sqrt'],
                                        'min_samples_leaf': [1, 2, 4,
5],
                                        'min_samples_split': [2, 3, 4,
5, 7,
                                                              10],
                                        'n_estimators': [400, 600,
800, 1000,
                                                         1200]},
                   random_state=42, verbose=2)
```

```
# Printing the best hyperparameters

rf_random.best_params_
```

```
{'n_estimators': 400,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': None,
 'bootstrap': False}
```

```python
# Checking accuracies on the train and test data, predicting on the
test data.

print('The best model is ', rf_random.best_estimator_)
print("\n The best model parameters are ",rf_random.best_params_)
print("\n The model accuracy on train set
is",rf_random.score(X_train,y_train))
print("\n The model accuracy on test set
is",rf_random.score(X_test,y_test))

y_predict=rf_random.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
The best model is  RandomForestClassifier(bootstrap=False,
n_estimators=400)

 The best model parameters are  {'n_estimators': 400,
'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'auto',
'max_depth': None, 'bootstrap': False}

 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9653179190751445


 Classification Report
              precision    recall  f1-score   support

         acc       0.95      0.91      0.93       118
        good       0.74      0.89      0.81        19
       unacc       0.99      0.99      0.99       358
       vgood       0.88      0.88      0.88        24

    accuracy                           0.97       519
   macro avg       0.89      0.92      0.90       519
weighted avg       0.97      0.97      0.97       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```
# Defining the model with the best parameters
clf = RandomForestClassifier(bootstrap=True, max_depth=None,
max_features='auto', min_samples_leaf=1, min_samples_split=3,
n_estimators = 1000)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier(min_samples_split=3, n_estimators=1000)
```

```
print("\n The model accuracy on train set
is",clf.score(X_train,y_train))
print("\n The model accuracy on test set is",clf.score(X_test,y_test))

y_predict=clf.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9633911368015414


 Classification Report
              precision    recall  f1-score   support

         acc       0.94      0.91      0.92       118
        good       0.77      0.89      0.83        19
```

```
       unacc       0.99      0.99      0.99       358
       vgood       0.84      0.88      0.86        24

    accuracy                           0.96       519
   macro avg       0.89      0.92      0.90       519
weighted avg       0.96      0.96      0.96       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```
# view the feature scores

feature_scores = pd.Series(clf.feature_importances_,
index=X_train.columns).sort_values(ascending=False)

feature_scores
```

```
safety      0.308498
persons     0.240509
maint       0.156148
buying      0.150323
lug_boot    0.085616
doors       0.058907
dtype: float64
```

```
# Creating a seaborn bar plot
```

```
sns.barplot(x=feature_scores, y=feature_scores.index)
# Add labels to the graph

plt.xlabel('Feature Importance Score')
plt.ylabel('Features')

# Add title to the graph
plt.title("Visualizing Important Features")

# Visualize the graph
plt.show()
```

Building a model on selected features without the least important feature

```
# declare feature vector and target variable, without the least
important feature

X1 = df.drop(['target', 'doors'], axis=1)

y1 = df['target']

# split data into training and testing sets

from sklearn.model_selection import train_test_split

X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1,
test_size = 0.33, random_state = 42)
```

```
# encode categorical variables with ordinal encoding

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'persons',
'lug_boot', 'safety'])


X_train1 = encoder.fit_transform(X_train1)

X_test1 = encoder.transform(X_test1)

# modelling without the least important feature
clf = RandomForestClassifier(bootstrap=True, max_depth=None,
max_features='auto', min_samples_leaf=1, min_samples_split=3,
n_estimators = 1000)
clf.fit(X_train1, y_train1)
```

```
RandomForestClassifier(min_samples_split=3, n_estimators=1000)
```

```
print("\n The model accuracy on train set
is",clf.score(X_train1,y_train1))
print("\n The model accuracy on test set
is",clf.score(X_test1,y_test1))

y_predict1=clf.predict(X_test1)
accuracy=accuracy_score(y_test1,y_predict1,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test1,y_predict1))
skplt.plot_confusion_matrix(y_test1,y_predict1)
```

```
 The model accuracy on train set is 0.9688850475367329

 The model accuracy on test set is 0.9334500875656743


 Classification Report
              precision    recall  f1-score   support

         acc       0.89      0.84      0.86       129
        good       0.56      0.90      0.69        20
       unacc       0.98      0.97      0.98       397
       vgood       0.80      0.80      0.80        25

    accuracy                           0.93       571
   macro avg       0.81      0.88      0.83       571
weighted avg       0.94      0.93      0.94       571


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

Xgboost

```
# import machine learning libraries
import xgboost as xgb

# import packages for hyperparameters tuning
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe

# Defining the feature space for search
space={'max_depth': hp.quniform("max_depth", 3, 18, 1),
        'gamma': hp.uniform ('gamma', 1,9),
        'reg_alpha' : hp.quniform('reg_alpha', 40,180,1),
        'reg_lambda' : hp.uniform('reg_lambda', 0,1),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0.5,1),
        'min_child_weight' : hp.quniform('min_child_weight', 0, 10,
1),
        'n_estimators': 180,
        'seed': 42
    }

# Deifning the objective function to minimize as this is required by
hyperopt library
def objective(space):
    clf=xgb.XGBClassifier(
                    n_estimators =space['n_estimators'], max_depth =
int(space['max_depth']), gamma = space['gamma'],
                    reg_alpha =
int(space['reg_alpha']),min_child_weight=int(space['min_child_weight']
```

```
),
                     colsample_bytree=int(space['colsample_bytree']))

    evaluation = [( X_train, y_train), ( X_test, y_test)]

    clf.fit(X_train, y_train,
            eval_set=evaluation, eval_metric = 'merror',
            early_stopping_rounds=10,verbose=2)


    pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, pred)
    print ("SCORE:", accuracy)
    return {'loss': -accuracy, 'status': STATUS_OK }

# Getting the optimum
trials = Trials()

best_hyperparams = fmin(fn = objective,
                        space = space,
                        algo = tpe.suggest,
                        max_evals = 100,
                        trials = trials)
```

```
   0%|
| 0/100 [00:00<?, ?trial/s, best loss=?]
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

[4]   validation_0-merror:0.29529validation_1-merror:0.31021

[6]   validation_0-merror:0.29529validation_1-merror:0.31021

[8]   validation_0-merror:0.29529validation_1-merror:0.31021

[10]  validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

  2%|▉                                                  | 2/100
[00:00<00:30,  3.26trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


SCORE:
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

  4%|▉                                                  | 4/100
[00:01<00:16,  5.80trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
```

```
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[9]  validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

  6%|██                                                  | 6/100
[00:01<00:12,  7.62trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
```

```
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

  8%|████                                                | 8/100
[00:01<00:10,  8.57trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
```

```
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
```

```
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


0.6897880539499036
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 13%|▉▉▉▉▉▉                                                | 13/100
[00:01<00:09,  9.66trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
```

```
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 15%|╸╸╸╸╸╸╸                                           | 15/100
[00:01<00:08,  9.93trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
```

```
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]  validation_0-merror:0.29529validation_1-merror:0.31021

[2]  validation_0-merror:0.29529validation_1-merror:0.31021

[4]  validation_0-merror:0.29529validation_1-merror:0.31021

[6]  validation_0-merror:0.29529validation_1-merror:0.31021

[8]  validation_0-merror:0.29529validation_1-merror:0.31021

[10] validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

 17%|┃┃┃┃┃┃┃┃                                          | 17/100
[00:02<00:08, 10.12trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
```

```
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 19%|╫╫╫╫╫╫╫╫╫                                          | 19/100
[00:02<00:07, 10.38trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
```

```
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

 23%|██████████                                        | 23/100
[00:02<00:08,  8.78trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

 25%|╫╫╫╫╫╫╫╫╫╫╫╫                                        | 25/100
[00:03<00:09,  8.09trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

 27%|╫╫╫╫╫╫╫╫╫╫╫╫                                        | 27/100
[00:03<00:09,  7.73trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

 29%|[redacted]| 29/100
[00:03<00:09,  7.71trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 31%|▉▉▉
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 33%|└└└
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
 37%|                                                  | 37/100
[00:04<00:10,  6.07trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

[4]   validation_0-merror:0.29529validation_1-merror:0.31021

[6]   validation_0-merror:0.29529validation_1-merror:0.31021

[8]   validation_0-merror:0.29529validation_1-merror:0.31021

[9]   validation_0-merror:0.29529validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]   validation_0-merror:0.29529validation_1-merror:0.31021

[2]   validation_0-merror:0.29529validation_1-merror:0.31021

 38%|                                                  | 38/100
[00:05<00:10,  6.19trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 40%|╫╫╫
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 44%|                                                  | 44/100
[00:06<00:09,  5.96trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 45%|▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉                                  | 45/100
[00:06<00:11,  4.95trial/s, best loss: -0.6897880539499036]


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 46%|▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▉▌                                 | 46/100
[00:06<00:10,  5.34trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
```

```
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

 48%|██████████████████                                | 48/100
[00:06<00:09,  5.70trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 50%|██████████████████████                                  | 50/100
[00:07<00:09,  5.52trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
```

```
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 52%|██████████████████████                                  | 52/100
[00:07<00:08,  5.67trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
```

```
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 54%|╫╫╫
```

```
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 55%|╫╫╫
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

 57%|██████████████████████                    | 57/100
[00:08<00:07,  5.99trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


0.6897880539499036
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 59%|██████████████████████████                    | 59/100
[00:09<00:07,  5.25trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 61%|██████████████████████████                  | 61/100
[00:09<00:10,  3.67trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

 61%|██████████████████████████                  | 61/100
[00:09<00:10,  3.67trial/s, best loss: -0.6897880539499036]
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 63%|██████████████████████████                  | 63/100
[00:10<00:10,  3.46trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 64%|└└└
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

 64%|██████████████████████████                  | 64/100
[00:10<00:09,  3.69trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 66%|                                                  | 66/100
[00:11<00:08,  3.78trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[4]	validation_0-merror:0.29529	validation_1-merror:0.31021
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 70%|□□□
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 72%|□□□
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 74%|▉▉▉
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

 78%|▉▉▉
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 81%|██████████████████████████████████          | 81/100
[00:13<00:04,  4.14trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
```

```
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 83%|╫╫╫
```

```
[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

 83%|┃┃┃
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 86%|██████████████████████████████████████       | 86/100
[00:15<00:03,  4.05trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 88%|▉▉▉
```

```
SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

 90%|▉▉▉
```

```
 91%|                                                    | 91/100
[00:16<00:02,  3.88trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021
```

```
 93%|▌▌▌
```

```
 95%|                                                  | 95/100
[00:17<00:00,  5.25trial/s, best loss: -0.6897880539499036]

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[10]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036
```

```
[0]	validation_0-merror:0.29529	validation_1-merror:0.31021

[2]	validation_0-merror:0.29529	validation_1-merror:0.31021

 97%|▉▉▉
```

```
0.6897880539499036

[0]	validation_0-merror:0.29529	validation_1-merror:0.31021



C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)


[2]	validation_0-merror:0.29529	validation_1-merror:0.31021
[4]	validation_0-merror:0.29529	validation_1-merror:0.31021

[6]	validation_0-merror:0.29529	validation_1-merror:0.31021

[8]	validation_0-merror:0.29529	validation_1-merror:0.31021

[9]	validation_0-merror:0.29529	validation_1-merror:0.31021

SCORE:

0.6897880539499036

100%|██████████████████████████████████████████| 100/100
[00:18<00:00,  5.55trial/s, best loss: -0.6897880539499036]
```

```python
# Getting the best model
print("The best hyperparameters are : ","\n")
print(best_hyperparams)
```

```
The best hyperparameters are :

{'colsample_bytree': 0.5356402785750933, 'gamma': 8.930986603401166,
'max_depth': 9.0, 'min_child_weight': 1.0, 'reg_alpha': 119.0,
'reg_lambda': 0.09812781449273855}
```

```python
# Best model
clf=xgb.XGBClassifier(best_hyperparams)
```

```
evaluation = [( X_train, y_train), ( X_test, y_test)]
clf.fit(X_train, y_train,
            eval_set=evaluation,
            early_stopping_rounds=10,verbose=2)
```

```
[08:15:56] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.
[0]	validation_0-mlogloss:0.94153	validation_1-mlogloss:0.94379
[2]	validation_0-mlogloss:0.53773	validation_1-mlogloss:0.55620
[4]	validation_0-mlogloss:0.34786	validation_1-mlogloss:0.37244
[6]	validation_0-mlogloss:0.23741	validation_1-mlogloss:0.26699
[8]	validation_0-mlogloss:0.17458	validation_1-mlogloss:0.21000
[10]	validation_0-mlogloss:0.13339	validation_1-mlogloss:0.17272
[12]	validation_0-mlogloss:0.10732	validation_1-mlogloss:0.14942
[14]	validation_0-mlogloss:0.08744	validation_1-mlogloss:0.12898
```

```
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\core.py:502:
FutureWarning: Pass `objective` as keyword args.  Passing these as
positional arguments will be considered as error in future releases.
  format(", ".join(args_msg)), FutureWarning
C:\Users\hp\Anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[16]	validation_0-mlogloss:0.07224	validation_1-mlogloss:0.11262
[18]	validation_0-mlogloss:0.06176	validation_1-mlogloss:0.10467
[20]	validation_0-mlogloss:0.05197	validation_1-mlogloss:0.09462
[22]	validation_0-mlogloss:0.04259	validation_1-mlogloss:0.08466
[24]	validation_0-mlogloss:0.03656	validation_1-mlogloss:0.08024
[26]	validation_0-mlogloss:0.03204	validation_1-mlogloss:0.07596
[28]	validation_0-mlogloss:0.02850	validation_1-mlogloss:0.07431
[30]	validation_0-mlogloss:0.02557	validation_1-mlogloss:0.07131
[32]	validation_0-mlogloss:0.02324	validation_1-mlogloss:0.07019
[34]	validation_0-mlogloss:0.02097	validation_1-mlogloss:0.06757
[36]	validation_0-mlogloss:0.01961	validation_1-mlogloss:0.06629
[38]	validation_0-mlogloss:0.01840	validation_1-mlogloss:0.06548
[40]	validation_0-mlogloss:0.01672	validation_1-mlogloss:0.06358
[42]	validation_0-mlogloss:0.01589	validation_1-mlogloss:0.06396
[44]	validation_0-mlogloss:0.01541	validation_1-mlogloss:0.06414
[46]	validation_0-mlogloss:0.01478	validation_1-mlogloss:0.06455
[48]	validation_0-mlogloss:0.01420	validation_1-mlogloss:0.06383
[50]	validation_0-mlogloss:0.01369	validation_1-mlogloss:0.06387
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1,
enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1,
missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, objective='multi:softprob',
predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1,
scale_pos_weight=None,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

```
print("\n The model accuracy on train set
is",clf.score(X_train,y_train))
print("\n The model accuracy on test set is",clf.score(X_test,y_test))

y_predict=clf.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9807321772639692


 Classification Report
              precision    recall  f1-score   support

         acc       0.97      0.95      0.96       118
        good       0.77      0.89      0.83        19
       unacc       1.00      1.00      1.00       358
       vgood       0.92      0.96      0.94        24

    accuracy                           0.98       519
   macro avg       0.92      0.95      0.93       519
weighted avg       0.98      0.98      0.98       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

Updates

```
import category_encoders as ce
encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors',
'persons', 'lug_boot', 'safety'])
df = encoder.fit_transform(df)

from sklearn.preprocessing import PolynomialFeatures

# summarize the shape of the dataset
print(df.shape)
# summarize each variable
display(df.head(2))

# perform a polynomial features transform of the dataset
trans = PolynomialFeatures(degree=4)
data = trans.fit_transform(df.iloc[:,:-1])

# convert the array back to a dataframe
dataset = pd.DataFrame(data)

# summarize
display(dataset.head(2))
```

```
(1728, 7)
```

```
   buying  maint  doors  persons  lug_boot  safety target
0       1      1      1        1         1       1  unacc
1       1      1      1        1         1       2  unacc
```

```
   0    1    2    3    4    5    6    7    8    9    ...  200  201
202  203  \
0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0
1.0  1.0
1  1.0  1.0  1.0  1.0  1.0  1.0  2.0  1.0  1.0  1.0  ...  4.0  1.0
2.0  4.0

   204  205  206  207  208   209
0  1.0  1.0  1.0  1.0  1.0   1.0
1  8.0  1.0  2.0  4.0  8.0  16.0

[2 rows x 210 columns]
```

```python
# Getting X and y
X = dataset
y = df['target']

X.head()
```

```
   0    1    2    3    4    5    6    7    8    9    ...  200  201
202  203  \
0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0
1.0  1.0
1  1.0  1.0  1.0  1.0  1.0  1.0  2.0  1.0  1.0  1.0  ...  4.0  1.0
2.0  4.0
2  1.0  1.0  1.0  1.0  1.0  1.0  3.0  1.0  1.0  1.0  ...  9.0  1.0
3.0  9.0
3  1.0  1.0  1.0  1.0  1.0  2.0  1.0  1.0  1.0  1.0  ...  1.0  8.0
4.0  2.0
4  1.0  1.0  1.0  1.0  1.0  2.0  2.0  1.0  1.0  1.0  ...  4.0  8.0
8.0  8.0

    204   205   206   207   208   209
0   1.0   1.0   1.0   1.0   1.0   1.0
1   8.0   1.0   2.0   4.0   8.0  16.0
2  27.0   1.0   3.0   9.0  27.0  81.0
3   1.0  16.0   8.0   4.0   2.0   1.0
4   8.0  16.0  16.0  16.0  16.0  16.0

[5 rows x 210 columns]
```

```python
# importing necessary packages
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
import scikitplot.metrics as skplt
from sklearn import tree
from sklearn.model_selection import train_test_split
```

```
# Splitting data into test train, using a 0.3 split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Exploring class distribution under train ,crossvalidation dataset
print('Training Dataset',X_train.shape,y_train.shape)
print('\n Class label distribution in Training Set\
n',y_train.value_counts())
print('\n***********')
print("\n CrossValidation Dataset",X_test.shape,y_test.shape)
print('\nClass label distribution in Cross Validation Set\
n',y_test.value_counts())
print('\n***********')
```

```
Training Dataset (1209, 210) (1209,)

 Class label distribution in Training Set
 unacc    852
acc      266
good      50
vgood     41
Name: target, dtype: int64

***********

 CrossValidation Dataset (519, 210) (519,)

Class label distribution in Cross Validation Set
 unacc    358
acc      118
vgood     24
good      19
Name: target, dtype: int64

***********
```

```
# Importing and defining the random search feature space.
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in range(400,1400,200)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(2, 15, num = 1)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [ 2,3, 4, 5, 7,10]
# Minimum number of samples required at each leaf node
```

```
min_samples_leaf = [1, 2, 4, 5]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [400, 600, 800, 1000, 1200], 'max_features': ['auto',
'sqrt'], 'max_depth': [2, None], 'min_samples_split': [2, 3, 4, 5, 7,
10], 'min_samples_leaf': [1, 2, 4, 5], 'bootstrap': [True, False]}
```

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available
cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions =
random_grid, n_iter = 50, cv = 3, verbose=2, random_state=42, n_jobs =
-1)
# Fit the random search model
rf_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
```

```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(),
n_iter=50,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [2, None],
                                        'max_features': ['auto',
'sqrt'],
                                        'min_samples_leaf': [1, 2, 4,
5],
                                        'min_samples_split': [2, 3, 4,
5, 7,
                                                              10],
                                        'n_estimators': [400, 600,
800, 1000,
                                                         1200]},
                   random_state=42, verbose=2)
```

```
# Printing the best hyperparameters

rf_random.best_params_
```

```
{'n_estimators': 1000,
 'min_samples_split': 3,
 'min_samples_leaf': 2,
 'max_features': 'sqrt',
 'max_depth': None,
 'bootstrap': False}
```

```
# Checking accuracies on the train and test data, predicting on the
test data.

print('The best model is ', rf_random.best_estimator_)
print("\n The best model parameters are ",rf_random.best_params_)
print("\n The model accuracy on train set
is",rf_random.score(X_train,y_train))
print("\n The model accuracy on test set
is",rf_random.score(X_test,y_test))

y_predict=rf_random.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
The best model is  RandomForestClassifier(bootstrap=False,
max_features='sqrt', min_samples_leaf=2,
                       min_samples_split=3, n_estimators=1000)

 The best model parameters are  {'n_estimators': 1000,
'min_samples_split': 3, 'min_samples_leaf': 2, 'max_features': 'sqrt',
'max_depth': None, 'bootstrap': False}

 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9845857418111753


 Classification Report
              precision    recall  f1-score   support

         acc       0.97      0.97      0.97       118
        good       0.90      1.00      0.95        19
       unacc       1.00      0.99      1.00       358
       vgood       0.92      0.92      0.92        24

    accuracy                           0.98       519
   macro avg       0.95      0.97      0.96       519
weighted avg       0.98      0.98      0.98       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```
# Defining the model with the best parameters
clf = RandomForestClassifier(bootstrap=True, max_depth=None,
max_features='auto', min_samples_leaf=1, min_samples_split=3,
n_estimators = 1000)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier(min_samples_split=3, n_estimators=1000)
```

```
print("\n The model accuracy on train set
is",clf.score(X_train,y_train))
print("\n The model accuracy on test set is",clf.score(X_test,y_test))

y_predict=clf.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9845857418111753


 Classification Report
              precision    recall  f1-score   support

         acc       0.97      0.97      0.97       118
        good       0.90      1.00      0.95        19
```

```
       unacc       1.00      0.99      1.00       358
       vgood       0.96      0.92      0.94        24

    accuracy                           0.98       519
   macro avg       0.96      0.97      0.96       519
weighted avg       0.98      0.98      0.98       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

```
# view the feature scores

feature_scores = pd.Series(clf.feature_importances_,
index=X_train.columns).sort_values(ascending=False)

feature_scores

24     0.035130
200    0.032737
119    0.027746
203    0.024956
135    0.020424
         ...
3      0.000210
175    0.000153
64     0.000112
18     0.000102
```

```
0      0.000000
Length: 210, dtype: float64
```

```
# Creating a seaborn bar plot

sns.barplot(x=feature_scores, y=feature_scores.index)
# Add labels to the graph

plt.xlabel('Feature Importance Score')
plt.ylabel('Features')

# Add title to the graph
plt.title("Visualizing Important Features")

# Visualize the graph
plt.show()
```

```
list(feature_scores[feature_scores < 0.01].index)
```

```
[109,
 37,
 151,
 82,
 190,
 114,
 208,
 70,
 169,
```

```
187,
134,
108,
204,
105,
36,
92,
63,
107,
48,
115,
113,
126,
47,
103,
104,
117,
196,
174,
139,
137,
130,
202,
112,
62,
181,
192,
207,
165,
4,
189,
185,
172,
12,
128,
164,
26,
91,
153,
163,
81,
43,
161,
206,
188,
106,
195,
22,
72,
```

```
75,
129,
74,
154,
33,
127,
35,
186,
209,
53,
183,
17,
191,
194,
97,
71,
149,
162,
89,
51,
31,
27,
110,
73,
6,
166,
83,
88,
131,
58,
15,
32,
98,
111,
148,
125,
150,
180,
144,
143,
10,
59,
100,
52,
87,
96,
69,
147,
182,
```

```
168,
66,
41,
157,
68,
42,
102,
124,
57,
133,
160,
142,
61,
122,
159,
99,
23,
56,
44,
20,
11,
46,
136,
146,
77,
177,
184,
16,
86,
141,
198,
28,
171,
50,
123,
7,
152,
201,
2,
30,
40,
121,
49,
140,
1,
55,
84,
179,
158,
```

```
 21,
 19,
 156,
 67,
 54,
 176,
 120,
 9,
 95,
 205,
 178,
 65,
 145,
 14,
 39,
 80,
 13,
 155,
 5,
 25,
 3,
 175,
 64,
 18,
 0]
```

Building a model on selected features without the least important feature

```
# declare feature vector and target variable, without the least
important feature

X1 = X.drop(list(feature_scores[feature_scores < 0.01].index), axis=1)

y1 = df['target']

# split data into training and testing sets

from sklearn.model_selection import train_test_split

X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1,
test_size = 0.33, random_state = 42)

# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available
cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions =
random_grid, n_iter = 50, cv = 3, verbose=2, random_state=42, n_jobs =
```

```
-1)
# Fit the random search model
rf_random.fit(X_train1, y_train1)
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
```

```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(),
n_iter=50,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [2, None],
                                        'max_features': ['auto',
'sqrt'],
                                        'min_samples_leaf': [1, 2, 4,
5],
                                        'min_samples_split': [2, 3, 4,
5, 7,
                                                              10],
                                        'n_estimators': [400, 600,
800, 1000,
                                                         1200]},
                   random_state=42, verbose=2)
```

```
# Printing the best hyperparameters

rf_random.best_params_
```

```
{'n_estimators': 600,
 'min_samples_split': 7,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': None,
 'bootstrap': True}
```

```
# Checking accuracies on the train and test data, predicting on the
test data.

print('The best model is ', rf_random.best_estimator_)
print("\n The best model parameters are ",rf_random.best_params_)
print("\n The model accuracy on train set
is",rf_random.score(X_train1,y_train1))
print("\n The model accuracy on test set
is",rf_random.score(X_test1,y_test1))

y_predict=rf_random.predict(X_test1)
accuracy=accuracy_score(y_test1,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test1,y_predict))
skplt.plot_confusion_matrix(y_test1,y_predict)
```

```
The best model is  RandomForestClassifier(min_samples_split=7,
n_estimators=600)

 The best model parameters are  {'n_estimators': 600,
'min_samples_split': 7, 'min_samples_leaf': 1, 'max_features': 'auto',
'max_depth': None, 'bootstrap': True}

 The model accuracy on train set is 0.9904926534140017

 The model accuracy on test set is 0.9597197898423818


 Classification Report
              precision    recall  f1-score   support

         acc       0.91      0.91      0.91       129
        good       0.70      0.95      0.81        20
       unacc       1.00      0.98      0.99       397
       vgood       0.85      0.92      0.88        25

    accuracy                           0.96       571
   macro avg       0.87      0.94      0.90       571
weighted avg       0.96      0.96      0.96       571


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

Using SMOTE for Upsampling

```
X_train

      0    1    2    3    4    5    6     7     8     9    ...   200
201  \
1178  1.0  3.0  3.0  4.0  2.0  3.0  3.0   9.0   9.0  12.0  ...  36.0
54.0
585   1.0  2.0  2.0  2.0  3.0  1.0  1.0   4.0   4.0   4.0  ...   9.0
3.0
1552  1.0  4.0  3.0  2.0  2.0  2.0  2.0  16.0  12.0   8.0  ...  16.0
16.0
1169  1.0  3.0  3.0  4.0  1.0  3.0  3.0   9.0   9.0  12.0  ...   9.0
27.0
1033  1.0  3.0  2.0  3.0  1.0  3.0  2.0   9.0   6.0   9.0  ...   4.0
27.0
...   ...  ...  ...  ...  ...  ...  ...   ...   ...   ...  ...   ...
...
1130  1.0  3.0  3.0  2.0  3.0  2.0  3.0   9.0   9.0   6.0  ...  81.0
24.0
1294  1.0  3.0  4.0  4.0  3.0  3.0  2.0   9.0  12.0  12.0  ...  36.0
81.0
860   1.0  2.0  4.0  4.0  3.0  2.0  3.0   4.0   8.0   8.0  ...  81.0
24.0
1459  1.0  4.0  2.0  3.0  1.0  1.0  2.0  16.0   8.0  12.0  ...   4.0
1.0
1126  1.0  3.0  3.0  2.0  3.0  1.0  2.0   9.0   9.0   6.0  ...  36.0
3.0

       202   203   204   205   206   207   208   209
1178  54.0  54.0  54.0  81.0  81.0  81.0  81.0  81.0
585    3.0   3.0   3.0   1.0   1.0   1.0   1.0   1.0
1552  16.0  16.0  16.0  16.0  16.0  16.0  16.0  16.0
1169  27.0  27.0  27.0  81.0  81.0  81.0  81.0  81.0
1033  18.0  12.0   8.0  81.0  54.0  36.0  24.0  16.0
...    ...   ...   ...   ...   ...   ...   ...   ...
1130  36.0  54.0  81.0  16.0  24.0  36.0  54.0  81.0
1294  54.0  36.0  24.0  81.0  54.0  36.0  24.0  16.0
860   36.0  54.0  81.0  16.0  24.0  36.0  54.0  81.0
1459   2.0   4.0   8.0   1.0   2.0   4.0   8.0  16.0
1126   6.0  12.0  24.0   1.0   2.0   4.0   8.0  16.0

[1209 rows x 210 columns]

y_train

1178    vgood
585     unacc
1552      acc
1169    unacc
```

```
1033    unacc
        ...
1130    vgood
1294     good
860       acc
1459    unacc
1126      acc
Name: target, Length: 1209, dtype: object
```

```python
from imblearn.over_sampling import SMOTE
from collections import Counter
from matplotlib import pyplot
from sklearn.preprocessing import LabelEncoder

# label encode the target variable
y = LabelEncoder().fit_transform(y_train)
# transform the dataset
oversample = SMOTE()
X, y = oversample.fit_resample(X_train, y)
# summarize distribution
counter = Counter(y)
for k,v in counter.items():
      per = v / len(y) * 100
      print('Class=%d, n=%d (%.3f%%)' % (k, v, per))
# plot the distribution
pyplot.bar(counter.keys(), counter.values())
pyplot.show()
```

```
Class=3, n=852 (25.000%)
Class=2, n=852 (25.000%)
Class=0, n=852 (25.000%)
Class=1, n=852 (25.000%)
```

```
X_train = X
y_train = y

y_test = LabelEncoder().fit_transform(y_test)

# Importing and defining the random search feature space.
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import RandomizedSearchCV

# Number of trees in random forest
n_estimators = [int(x) for x in range(400,1400,200)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(2, 15, num = 1)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [ 2,3, 4, 5, 7,10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4, 5]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
```

```
                'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [400, 600, 800, 1000, 1200], 'max_features': ['auto',
'sqrt'], 'max_depth': [2, None], 'min_samples_split': [2, 3, 4, 5, 7,
10], 'min_samples_leaf': [1, 2, 4, 5], 'bootstrap': [True, False]}
```

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available
cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions =
random_grid, n_iter = 30, cv = 3, verbose=2, random_state=42, n_jobs =
-1)
# Fit the random search model
rf_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 30 candidates, totalling 90 fits
```

```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(),
n_iter=30,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [2, None],
                                        'max_features': ['auto',
'sqrt'],
                                        'min_samples_leaf': [1, 2, 4,
5],
                                        'min_samples_split': [2, 3, 4,
5, 7,
                                                              10],
                                        'n_estimators': [400, 600,
800, 1000,
                                                         1200]},
                   random_state=42, verbose=2)
```

```
# Printing the best hyperparameters

rf_random.best_params_
```

```
{'n_estimators': 600,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': None,
 'bootstrap': False}
```

```
# Checking accuracies on the train and test data, predicting on the
test data.
```

```
print('The best model is ', rf_random.best_estimator_)
print("\n The best model parameters are ",rf_random.best_params_)
print("\n The model accuracy on train set
is",rf_random.score(X_train,y_train))
print("\n The model accuracy on test set
is",rf_random.score(X_test,y_test))

y_predict=rf_random.predict(X_test)
accuracy=accuracy_score(y_test,y_predict,normalize=True)*float(100)
print('\n\n Classification Report')
print(classification_report(y_test,y_predict))
skplt.plot_confusion_matrix(y_test,y_predict)
```

```
The best model is  RandomForestClassifier(bootstrap=False,
min_samples_split=5, n_estimators=600)

 The best model parameters are  {'n_estimators': 600,
'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'auto',
'max_depth': None, 'bootstrap': False}

 The model accuracy on train set is 1.0

 The model accuracy on test set is 0.9865125240847784


 Classification Report
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       118
           1       0.86      1.00      0.93        19
           2       1.00      0.99      1.00       358
           3       0.96      1.00      0.98        24

    accuracy                           0.99       519
   macro avg       0.95      0.99      0.97       519
weighted avg       0.99      0.99      0.99       519


<AxesSubplot:title={'center':'Confusion Matrix'}, xlabel='Predicted
label', ylabel='True label'>
```

Confusion Matrix
True label
Predicted label
0
1
2
3
114
3
0
1
0
19
0
0
3
0
355
0
0
0
0
24
350
300
250
200
150
100
50
0

```python
# Importing the required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')

import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# import SVC classifier
from sklearn.svm import SVC

# Hyperparameter tunning libraries
from sklearn.model_selection import RandomizedSearchCV

# import metrics to compute accuracy
from sklearn.metrics import accuracy_score, f1_score,
classification_report, confusion_matrix, roc_curve, roc_auc_score

# Reading the data
df_train = pd.read_csv('pulsar_data_train.csv')
df_test = pd.read_csv('pulsar_data_test.csv')

# Visualizing the dataframe
df_train.head()
```

```
    Mean of the integrated profile  \
0                       121.156250
1                        76.968750
2                       130.585938
3                       156.398438
4                        84.804688

    Standard deviation of the integrated profile  \
0                                      48.372971
1                                      36.175557
2                                      53.229534
3                                      48.865942
4                                      36.117659

    Excess kurtosis of the integrated profile  \
0                                    0.375485
1                                    0.712898
2                                    0.133408
3                                   -0.215989
4                                    0.825013

    Skewness of the integrated profile   Mean of the DM-SNR curve  \
0                            -0.013165                   3.168896
```

```
1                              3.388719                    2.399666
2                             -0.297242                    2.743311
3                             -0.171294                   17.471572
4                              3.274125                    2.790134

    Standard deviation of the DM-SNR curve  \
0                                18.399367
1                                17.570997
2                                22.362553
3                                      NaN
4                                20.618009

    Excess kurtosis of the DM-SNR curve   Skewness of the DM-SNR curve
\
0                               7.449874                       65.159298

1                               9.414652                      102.722975

2                               8.508364                       74.031324

3                               2.958066                        7.197842

4                               8.405008                       76.291128


   target_class
0           0.0
1           0.0
2           0.0
3           0.0
4           0.0

df_test.head()

    Mean of the integrated profile  \
0                       116.906250
1                        75.585938
2                       103.273438
3                       101.078125
4                       113.226562

    Standard deviation of the integrated profile  \
0                                       48.920605
1                                       34.386254
2                                       46.996628
3                                       48.587487
4                                       48.608804

    Excess kurtosis of the integrated profile  \
0                                    0.186046
1                                    2.025498
```

```
2                                     0.504295
3                                     1.011427
4                                     0.291538

    Skewness of the integrated profile   Mean of the DM-SNR curve  \
0                            -0.129815                   3.037625
1                             8.652913                   3.765050
2                             0.821088                   2.244983
3                             1.151870                  81.887960
4                             0.292120                   6.291806

    Standard deviation of the DM-SNR curve  \
0                                17.737102
1                                21.897049
2                                15.622566
3                                81.464136
4                                26.585056

    Excess kurtosis of the DM-SNR curve   Skewness of the DM-SNR curve
\
0                              8.122621                      78.813405

1                              7.048189                      55.878791

2                              9.330498                     105.134941

3                              0.485105                      -1.117904

4                              4.540138                      21.708268


   target_class
0           NaN
1           NaN
2           NaN
3           NaN
4           NaN
```

```
# Checking the shape of the data
df_train.shape
```

```
(12528, 9)
```

```
df_test.shape
```

```
(5370, 9)
```

```
#Checking for null values and data types
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12528 entries, 0 to 12527
Data columns (total 9 columns):
 #   Column                                             Non-Null Count
Dtype
---  ------                                             --------------
-----
 0    Mean of the integrated profile                    12528 non-null
float64
 1    Standard deviation of the integrated profile  12528 non-null
float64
 2    Excess kurtosis of the integrated profile     10793 non-null
float64
 3    Skewness of the integrated profile            12528 non-null
float64
 4    Mean of the DM-SNR curve                      12528 non-null
float64
 5    Standard deviation of the DM-SNR curve        11350 non-null
float64
 6    Excess kurtosis of the DM-SNR curve           12528 non-null
float64
 7    Skewness of the DM-SNR curve                  11903 non-null
float64
 8   target_class                                   12528 non-null
float64
dtypes: float64(9)
memory usage: 881.0 KB
```

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5370 entries, 0 to 5369
Data columns (total 9 columns):
 #   Column                                             Non-Null Count
Dtype
---  ------                                             --------------
-----
 0    Mean of the integrated profile                    5370 non-null
float64
 1    Standard deviation of the integrated profile  5370 non-null
float64
 2    Excess kurtosis of the integrated profile     4603 non-null
float64
 3    Skewness of the integrated profile            5370 non-null
float64
 4    Mean of the DM-SNR curve                      5370 non-null
float64
 5    Standard deviation of the DM-SNR curve        4846 non-null
float64
 6    Excess kurtosis of the DM-SNR curve           5370 non-null
```

```
float64
 7    Skewness of the DM-SNR curve                  5126 non-null
float64
 8   target_class                                   0 non-null
float64
dtypes: float64(9)
memory usage: 377.7 KB
```

```
# Null values
df_train.isnull().sum()
```

```
 Mean of the integrated profile                     0
 Standard deviation of the integrated profile       0
 Excess kurtosis of the integrated profile       1735
 Skewness of the integrated profile                 0
 Mean of the DM-SNR curve                           0
 Standard deviation of the DM-SNR curve          1178
 Excess kurtosis of the DM-SNR curve                0
 Skewness of the DM-SNR curve                     625
target_class                                        0
dtype: int64
```

```
# Null values
df_test.isnull().sum()
```

```
 Mean of the integrated profile                     0
 Standard deviation of the integrated profile       0
 Excess kurtosis of the integrated profile        767
 Skewness of the integrated profile                 0
 Mean of the DM-SNR curve                           0
 Standard deviation of the DM-SNR curve           524
 Excess kurtosis of the DM-SNR curve                0
 Skewness of the DM-SNR curve                     244
target_class                                     5370
dtype: int64
```

```
#description
df_train.describe().transpose()
```

```
                                                   count        mean  \
 Mean of the integrated profile                  12528.0  111.041841
 Standard deviation of the integrated profile  12528.0   46.521437
 Excess kurtosis of the integrated profile     10793.0    0.478548
 Skewness of the integrated profile            12528.0    1.778431
 Mean of the DM-SNR curve                      12528.0   12.674758
 Standard deviation of the DM-SNR curve        11350.0   26.351318
 Excess kurtosis of the DM-SNR curve           12528.0    8.333489
 Skewness of the DM-SNR curve                  11903.0  105.525779
target_class                                   12528.0    0.092034

                                                      std
```

```
min  \
 Mean of the integrated profile                 25.672828   5.812500

 Standard deviation of the integrated profile    6.801077  24.772042

 Excess kurtosis of the integrated profile       1.064708  -1.738021

 Skewness of the integrated profile              6.208450  -1.791886

 Mean of the DM-SNR curve                       29.613230   0.213211

 Standard deviation of the DM-SNR curve         19.610842   7.370432

 Excess kurtosis of the DM-SNR curve             4.535783  -3.139270

 Skewness of the DM-SNR curve                  107.399585  -1.976976

target_class                                     0.289085   0.000000


                                                       25%         50%
\
 Mean of the integrated profile                 100.871094  115.183594

 Standard deviation of the integrated profile   42.362222   46.931022

 Excess kurtosis of the integrated profile       0.024652    0.223678

 Skewness of the integrated profile             -0.188142    0.203317

 Mean of the DM-SNR curve                        1.910535    2.792642

 Standard deviation of the DM-SNR curve         14.404353   18.412402

 Excess kurtosis of the DM-SNR curve             5.803063    8.451097

 Skewness of the DM-SNR curve                   35.199899   83.126301

target_class                                     0.000000    0.000000


                                                       75%         max

 Mean of the integrated profile                 127.109375  189.734375

 Standard deviation of the integrated profile   50.979103   91.808628

 Excess kurtosis of the integrated profile       0.473125    8.069522

 Skewness of the integrated profile              0.932374   68.101622

 Mean of the DM-SNR curve                        5.413253  222.421405
```

```
 Standard deviation of the DM-SNR curve         28.337418   110.642211

 Excess kurtosis of the DM-SNR curve            10.727927    34.539844

 Skewness of the DM-SNR curve                  139.997850  1191.000837

target_class                                     0.000000     1.000000
```

```
#description
df_test.describe().transpose()
```

```
                                                 count        mean
std  \
 Mean of the integrated profile                 5370.0  111.168917
25.608635
 Standard deviation of the integrated profile  5370.0   46.615074
6.940638
 Excess kurtosis of the integrated profile     4603.0    0.483676
1.076893
 Skewness of the integrated profile            5370.0    1.751260
6.072820
 Mean of the DM-SNR curve                      5370.0   12.473587
29.145134
 Standard deviation of the DM-SNR curve        4846.0   26.425371
19.384489
 Excess kurtosis of the DM-SNR curve           5370.0    8.233724
4.435683
 Skewness of the DM-SNR curve                  5126.0  102.869088
104.748418
target_class                                      0.0         NaN
NaN

                                                       min
25%  \
 Mean of the integrated profile                  6.179688  101.041016

 Standard deviation of the integrated profile  24.791612   42.408020

 Excess kurtosis of the integrated profile     -1.876011    0.030643

 Skewness of the integrated profile            -1.764717   -0.189557

 Mean of the DM-SNR curve                       0.213211    1.956522

 Standard deviation of the DM-SNR curve         7.370432   14.555826

 Excess kurtosis of the DM-SNR curve           -2.721857    5.700461

 Skewness of the DM-SNR curve                  -1.964998   33.817330
```

```
target_class                                                NaN         NaN


                                                             50%         75%
\
 Mean of the integrated profile                     114.757812  127.023438

 Standard deviation of the integrated profile        47.031304   51.133444

 Excess kurtosis of the integrated profile            0.227314    0.475056

 Skewness of the integrated profile                   0.186468    0.918807

 Mean of the DM-SNR curve                             2.830686    5.590301

 Standard deviation of the DM-SNR curve              18.549670   28.681787

 Excess kurtosis of the DM-SNR curve                  8.383695   10.632265

 Skewness of the DM-SNR curve                        81.392046  136.893502

target_class                                                NaN         NaN


                                                            max
 Mean of the integrated profile                     192.617188
 Standard deviation of the integrated profile        98.778911
 Excess kurtosis of the integrated profile            7.608370
 Skewness of the integrated profile                  65.385974
 Mean of the DM-SNR curve                           223.392141
 Standard deviation of the DM-SNR curve             109.712649
 Excess kurtosis of the DM-SNR curve                 34.539844
 Skewness of the DM-SNR curve                       1191.000837
target_class                                                NaN
```

```
#Plotting target distribution
f,ax=plt.subplots(1,2,figsize=(18,8))
df_train['target_class'].value_counts().plot.pie(ax = ax[0],
explode=[0,0.1],autopct='%1.1f%%',shadow=False, textprops={'fontsize':
14})
ax[0].set_title('target')
ax[0].set_ylabel('')
sns.countplot('target_class',data=df_train,ax=ax[1])
ax[1].set_title('target')
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
```

```
error or misinterpretation.
  FutureWarning

Text(0.5, 1.0, 'target')
```

Preprocessing

```
# Checking the column names
df_train.columns

Index([' Mean of the integrated profile',
       ' Standard deviation of the integrated profile',
       ' Excess kurtosis of the integrated profile',
       ' Skewness of the integrated profile', ' Mean of the DM-SNR
curve',
       ' Standard deviation of the DM-SNR curve',
       ' Excess kurtosis of the DM-SNR curve', ' Skewness of the DM-
SNR curve',
       'target_class'],
      dtype='object')

# Getting the feature and target matrices
target = df_train["target_class"]
features = df_train.drop("target_class", axis=1)
```

EDA

```
# Creating histplots
fig, axes = plt.subplots(nrows=8, ncols=1, figsize=(4,20))
```

```
for i in range(len(features.columns)):
    col = features.columns[i]
    sns.histplot(x=df_train[col], hue=target, palette="cool",
kde=True, ax=axes[i])
plt.show()
```

target_class
0.0
1.0
Count
Mean of the integrated profile

target_class
0.0
1.0
Count
Standard deviation of the integrated profile

target_class
0.0
1.0
Count
Excess kurtosis of the integrated profile

target_class
0.0
1.0
Count
Skewness of the integrated profile

target_class
0.0
1.0
Count

```
# Pairplotusing seaborn
sns.pairplot(data=df_train,
             palette="husl",
             hue="target_class",
             vars=[" Mean of the integrated profile",
                   " Excess kurtosis of the integrated profile",
                   " Skewness of the integrated profile",
                   " Mean of the DM-SNR curve",
                   " Excess kurtosis of the DM-SNR curve",
                   " Skewness of the DM-SNR curve"])

plt.suptitle("PairPlot of Data Without Std. Dev. Fields",fontsize=18)

plt.tight_layout()
plt.show()
```

```
#Plotting the Correlation Heatmap
plt.figure(figsize=(16,12))
sns.heatmap(data=df_train.corr(),annot=True,cmap="bone",linewidths=1,f
mt=".2f",linecolor="gray")
plt.title("Correlation Map",fontsize=20)
plt.tight_layout()
plt.show()
```

Handling Missing Values

```
3 Computing the median for the three features with null values
median1 = df_train[' Excess kurtosis of the integrated
profile'].median()
median2 = df_train[' Standard deviation of the DM-SNR curve'].median()
median3 = df_train[' Skewness of the DM-SNR curve'].median()

# Imputing the null values
df_train[' Excess kurtosis of the integrated profile'] = df_train['
Excess kurtosis of the integrated profile'].fillna(median1)
```

```
df_test[' Excess kurtosis of the integrated profile'] = df_test['
Excess kurtosis of the integrated profile'].fillna(median1)

df_train[' Standard deviation of the DM-SNR curve'] = df_train['
Standard deviation of the DM-SNR curve'].fillna(median1)
df_test[' Standard deviation of the DM-SNR curve'] = df_test['
Standard deviation of the DM-SNR curve'].fillna(median1)

df_train[' Skewness of the DM-SNR curve'] = df_train[' Skewness of the
DM-SNR curve'].fillna(median1)
df_test[' Skewness of the DM-SNR curve'] = df_test[' Skewness of the
DM-SNR curve'].fillna(median1)

# Checking for null values again
df_train.isna().sum()
```

```
 Mean of the integrated profile                    0
 Standard deviation of the integrated profile      0
 Excess kurtosis of the integrated profile         0
 Skewness of the integrated profile                0
 Mean of the DM-SNR curve                          0
 Standard deviation of the DM-SNR curve            0
 Excess kurtosis of the DM-SNR curve               0
 Skewness of the DM-SNR curve                      0
target_class                                       0
dtype: int64
```

```
df_test.isna().sum()
```

```
 Mean of the integrated profile                      0
 Standard deviation of the integrated profile        0
 Excess kurtosis of the integrated profile           0
 Skewness of the integrated profile                  0
 Mean of the DM-SNR curve                            0
 Standard deviation of the DM-SNR curve              0
 Excess kurtosis of the DM-SNR curve                 0
 Skewness of the DM-SNR curve                        0
target_class                                      5370
dtype: int64
```

Modelling

```
# Splitting into test and train data
xtrain, xtest, ytrain, ytest = train_test_split(features, target,
test_size=0.2)

print(xtrain.shape)
print(xtest.shape)
```

```
(10022, 8)
(2506, 8)
```

```
# Using standard scalar to scale
ss = StandardScaler()
ss.fit(xtrain)
xtrain_array = ss.transform(xtrain)
xtest_array = ss.transform(xtest)

xtrain = pd.DataFrame(xtrain_array, columns = xtrain.columns)
xtest = pd.DataFrame(xtest_array, columns=xtest.columns)
```

Running SVM with default hyperparameters

Default hyperparameter means C=1.0, kernel=rbf and gamma=auto among other parameters.

```
xtrain.fillna(0, inplace=True)
xtest.fillna(0, inplace=True)

# instantiate classifier with default hyperparameters
svc=SVC()


# fit classifier to training set
svc.fit(xtrain,ytrain)


# make predictions on test set
y_pred=svc.predict(xtest)


# compute and print accuracy score
print('Model accuracy score with default hyperparameters: {0:0.4f}'.
format(accuracy_score(ytest, y_pred)))
print('Model f1 score with default hyperparameters: {0:0.4f}'.
format(f1_score(ytest, y_pred)))
```

```
Model accuracy score with default hyperparameters: 0.9741
Model f1 score with default hyperparameters: 0.8426
```

Using Random Search to find the best hyperparameters

```
# USing grid search
svc_clf = SVC()
params = [{
    'C':np.arange(0.01,10.0,0.1),
    'kernel':['linear','poly','rbf','sigmoid'],
    'degree':range(1,20),
    "gamma" : ['scale','auto'],
    'probability':[True],
```

```
    'class_weight':['balanced'],
    'random_state':[21],
    'max_iter':[-1],
}]


random_search_svc_clf = RandomizedSearchCV(svc_clf, scoring = 'f1',
param_distributions=params, n_jobs = -1, return_train_score = True,
n_iter=100, cv=2, random_state=21)
random_search_svc_clf.fit(xtrain,ytrain)
print("Best parameters scores:")
print(random_search_svc_clf.best_params_)
df = pd.DataFrame(random_search_svc_clf.cv_results_)
print("Mean Train Score:",
random_search_svc_clf.cv_results_['mean_train_score']
[df[df['mean_test_score']==random_search_svc_clf.best_score_].index]
[0])
print("Mean Validation score:", random_search_svc_clf.best_score_)
```

```
Best parameters scores:
{'random_state': 21, 'probability': True, 'max_iter': -1, 'kernel':
'rbf', 'gamma': 'auto', 'degree': 9, 'class_weight': 'balanced', 'C':
2.51}
Mean Train Score: 0.8691440763105074
Mean Validation score: 0.8572325581395349
```

Building the best model

```
clf = SVC(random_state=21, probability= True, max_iter= -1, kernel=
'rbf', gamma= 'auto', degree = 9, class_weight= 'balanced', C= 2.51)

# fit classifier to training set
clf.fit(xtrain,ytrain)


# make predictions on test set
y_pred=clf.predict(xtest)

cm = confusion_matrix(ytest, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[2231   46]
 [  34  195]]

True Positives(TP) =  2231

True Negatives(TN) =  195

False Positives(FP) =  46

False Negatives(FN) =  34
```

```
# visualize confusion matrix with seaborn heatmap

cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1',
'Actual Negative:0'],
                                 index=['Predict Positive:1', 'Predict
Negative:0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

```
<AxesSubplot:>
```

```
print(classification_report(ytest, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.98      0.98      0.98      2277
         1.0       0.81      0.85      0.83       229
```

```
    accuracy                           0.97      2506
   macro avg       0.90      0.92      0.91      2506
weighted avg       0.97      0.97      0.97      2506
```

```
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]

# print precision score

precision = TP / float(TP + FP)
print('Precision : {0:0.4f}'.format(precision))

recall = TP / float(TP + FN)
print('Recall or Sensitivity : {0:0.4f}'.format(recall))

true_positive_rate = TP / float(TP + FN)
print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))

false_positive_rate = FP / float(FP + TN)
print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))

specificity = TN / (TN + FP)
print('Specificity : {0:0.4f}'.format(specificity))
```

```
Precision : 0.9798
Recall or Sensitivity : 0.9850
True Positive Rate : 0.9850
False Positive Rate : 0.1909
Specificity : 0.8091
```

```
# plot ROC Curve

fpr, tpr, thresholds = roc_curve(ytest, y_pred)

plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, linewidth=2)

plt.plot([0,1], [0,1], 'k--' )

plt.rcParams['font.size'] = 12

plt.title('ROC curve for Predicting a Pulsar Star classifier')

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')
```

```
plt.show()
```

```
ROC_AUC = roc_auc_score(ytest, y_pred)
print('ROC AUC : {:.4f}'.format(ROC_AUC))
ROC AUC : 0.9157
```