**Contents:**

**ASSIGNMENT # 01**

[**Required ML Approach** 2](#_Toc140856713)

[**Loading Dataset** 2](#_Toc140856714)

[**Getting Data Information** 3](#_Toc140856715)

[**Data Observation** 3](#_Toc140856716)

[**Extracting Correlation** 4](#_Toc140856717)

[**Correlation Analysis** 5](#_Toc140856718)

[**Selecting the best 3 Models** 6](#_Toc140856719)

[**Linear Regression Model** 6](#_Toc140856720)

[**Model Performance** 7](#_Toc140856721)

[**Models Performance Evaluation** 8](#_Toc140856722)

[**Multivariate Regression** 9](#_Toc140856723)

[**Model With 3 Best Predictors** 9](#_Toc140856724)

[**Model With Other Predictors** 10](#_Toc140856725)

**ASSIGNMENT # 02**

[**Loading Dataset** 12](#_Toc140856726)

[**Data Observation** 13](#_Toc140856727)

[**Handling Categorical Data** 14](#_Toc140856728)

[**Splitting Datatest** 16](#_Toc140856729)

[**Feature Scaling** 16](#_Toc140856730)

[**ML Models** 17](#_Toc140856731)

[**Model Training** 18](#_Toc140856732)

[**Model Evaluation** 18](#_Toc140856733)

[**Model Comparision** 22](#_Toc140856734)

[**Finding Best Model** 23](#_Toc140856735)

[**Conclusion** 23](#_Toc140856736)

Assignment 1 (Medical Insurance)

**Q# 01: Is the required ML supervised, unsupervised, or semi-supervised learning and why? Which ML task (classification, clustering, regression analysis, or any other) is the best in this case and why?**

## **Required ML Approach**

The required ML approach for this task is supervised learning because we have labeled data with the target variable, “medicalcost”, indicating the medical expenses of individuals. In supervised learning, models are trained on labeled data to make predictions or estimates based on input features.

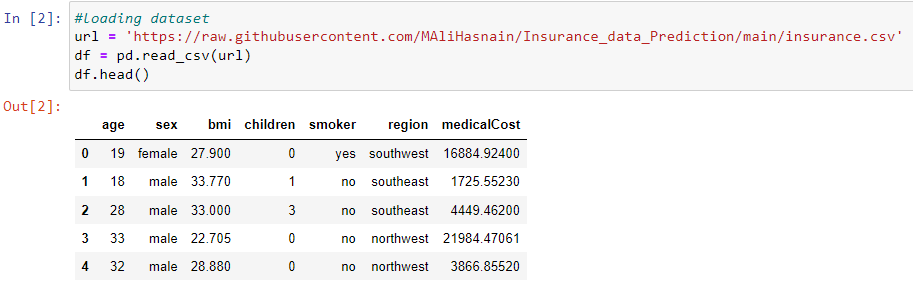
In this case, the best ML task is regression analysis. Regression analysis is used when the target variable is continuous or numeric, which is the scenario here. We aim to estimate the medical cost, which is a continuous variable, based on the available predictors such as age, sex, BMI, children, smoker, and region

By applying regression analysis, we can build a prediction model that can accurately estimate the medical cost of individuals using the provided predictors. This model will enable us to forecast the expected medical expenses based on factors like age, gender, BMI, number of children, smoking habits, and residential region.

The most suitable ML task for this case is regression analysis. This will allow us to develop a predictive model that estimates the medical cost of individuals by leveraging the available predictors, facilitating a better understanding and forecasting of medical expenses in the context of insurance.

## **Loading Dataset**

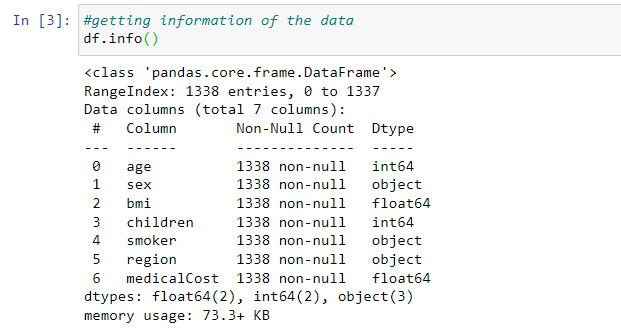
This code loads an insurance dataset from a specified URL using the pd.read\_csv() function. The dataset is stored in a DataFrame called df. The df.head() function is then used to display the first few rows of the dataset, providing a quick overview of the data's structure and contents. This code allows for easy access to the dataset and facilitates initial exploration and understanding of the data for further analysis.



**Q# 02: Explore your data and document your observation.**

## **Getting Data Information**

The code snippet df.info() is used to obtain information about the dataset stored in the DataFrame df. It provides a summary of the dataset's structure, including the number of rows and columns, the data type of each column, and the count of non-null values. This information is helpful in understanding the completeness and integrity of the dataset, identifying any missing values, and determining the data types of the variables. Including this code in the report allows for a concise overview of the dataset's properties and assists in the initial assessment of the data quality.



## **Data Observation**

we have a total of 1,338 entries (rows) and 7 columns. Here are our observations regarding the data:

**Age:** The "age" column represents the age of the primary insurance beneficiary. It is stored as an integer data type, and there are no missing values (non-null count: 1338). We can further analyze the distribution of ages to understand the age range and identify any potential outliers.

**Sex:** The "sex" column indicates the gender of the insurance contractor. It is stored as an object data type, which suggests it is a categorical variable. There are no missing values (non-null count: 1338). We can explore the distribution of genders to determine if there is any gender imbalance within the dataset.

**BMI:** The "bmi" column represents the body mass index of individuals. It is stored as a float data type and does not contain any missing values (non-null count: 1338). We can analyze the distribution of BMI values to understand the range and identify any potential outliers.

**Children:** The "children" column indicates the number of children covered by health insurance. It is stored as an integer data type, and there are no missing values (non-null count: 1338). We can explore the distribution of the number of children covered to identify any patterns or outliers.

**Smoker:** The "smoker" column specifies whether an individual is a smoker or not. It is stored as an object data type, implying it is a categorical variable. There are no missing values (non-null count: 1338). We can analyze the proportion of smokers and non-smokers within the dataset to understand the prevalence of smoking among the insured population.

**Region:** The "region" column represents the residential area of the payees in the US. It is stored as an object data type, indicating it is a categorical variable. There are no missing values (non-null count: 1338). We can examine the distribution of individuals across different regions to identify any regional patterns or differences.

**MedicalCost:** The "medicalCost" column is the target variable, representing the individual medical costs billed by the insurance company. It is stored as a float data type and does not contain any missing values (non-null count: 1338). We can analyze the distribution of medical costs to understand the range and identify any extreme values or outliers.

By exploring the dataset, we can gain insights into the characteristics of the data, such as age distribution, gender balance, BMI ranges, number of children covered, smoking prevalence, regional distribution, and medical cost distribution. These observations will guide us in further analysis and modeling to develop a prediction model for estimating medical costs based on the given predictors.

**Q#03: Study the correlation between each predictor and the medicalCost. What is your conclusion?**

## **Extracting Correlation**

Encoding and Correlation Analysis of Categorical Variables: This section of the analysis focuses on encoding the categorical variables 'sex' and 'smoker' into numeric values (0 and 1) and performing correlation analysis on the dataset. The code snippet begins by mapping 'female' to 0 and 'male' to 1 for the 'sex' variable, and 'no' to 0 and 'yes' to 1 for the 'smoker' variable. It then utilizes one-hot encoding to transform the categorical variable 'region' into separate binary dummy variables. Next, the code calculates the correlation coefficients between 'age', 'sex', 'bmi', 'children', 'smoker', and 'medicalCost'. To summarize the correlations related to the 'region' variable, the code filters and sums the correlation coefficients that include 'region\_'. Finally, the correlation coefficients are updated to include the merged region correlation, and the resulting correlation coefficient for 'medicalCost' is displayed. This analysis sheds light on the relationships between the predictors and the 'medicalCost' variable, providing insights into potential influences on medical expenses based on the given dataset.



## **Correlation Analysis**

**Age (0.299008):** The moderate positive correlation between age and medical costs indicates that there is a tendency for medical expenses to increase as individuals get older. This correlation suggests that age is a factor contributing to higher medical costs. Older individuals may require more frequent medical visits, specialized treatments, or ongoing management of chronic conditions, leading to higher healthcare expenses. It highlights the importance of considering age as a significant predictor when estimating medical costs.

**Sex (0.057292):** The weak positive correlation between sex and medical costs suggests that gender has a limited influence on medical expenses in the dataset. This correlation value indicates that there is a slight association between gender and medical costs, but the effect is relatively small. It implies that, overall, gender alone does not strongly impact the level of medical expenses. Other factors, such as specific health conditions or healthcare utilization patterns, may have a more substantial influence on medical costs than gender.

**BMI (0.198341):** The moderate positive correlation between BMI and medical costs indicates that a higher body mass index is associated with higher medical expenses. This correlation suggests that individuals with higher BMI values tend to incur more healthcare costs. Higher BMI is often associated with increased risks for chronic diseases, such as cardiovascular conditions, diabetes, and musculoskeletal issues. These conditions may require more extensive medical interventions, treatments, or ongoing management, leading to higher medical expenses.

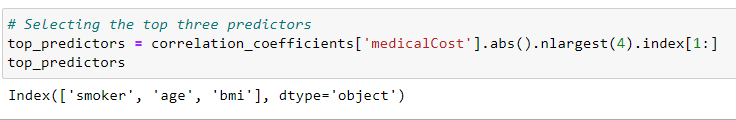
**Children (0.067998):** The weak positive correlation between the number of children covered by insurance and medical costs suggests that having more children covered by health insurance may slightly contribute to higher medical expenses. This correlation value indicates that there is a slight association between the number of children and medical costs. However, the effect is relatively small, implying that the number of children covered by insurance alone does not strongly impact the level of medical expenses. Other factors, such as specific health conditions or healthcare needs of the children, may play a more significant role in determining medical costs.

**Smoker (0.787251):** The strong positive correlation between smoking and medical costs reveals a significant association between these variables. This correlation suggests that smokers tend to have significantly higher medical expenses compared to non-smokers. Smoking is a well-known risk factor for various health conditions, such as lung cancer, cardiovascular diseases, and respiratory disorders. These conditions often require extensive medical care, including surgeries, medications, and ongoing treatments, leading to substantially higher healthcare costs for smokers compared to non-smokers.

It is important to note that correlation does not imply causation. While the correlations indicate the strength and direction of the relationship between each predictor and medical costs, they do not provide conclusive evidence of causation. Further analysis and consideration of other factors are necessary to establish causal relationships and to better understand the specific mechanisms behind the observed correlations.

**Q#04: Use the correlation analysis to select the 3 best predictors and build a simple linear regression model based on each of the predictors.**

## **Selecting the best 3 Models**

This code selects the top three predictors based on their correlation with the 'medicalCost' variable. The code calculates the absolute correlation coefficients between each predictor and the target variable and identifies the indices of the top four predictors with the largest coefficients, excluding the 'medicalCost' variable itself. The resulting output reveals that the predictors 'smoker', 'age', and 'BMI' have the highest correlation coefficients. These predictors exhibit stronger associations with medical costs compared to other variables in the dataset. This information is crucial for further analysis and model development, as it highlights the potential significance of these predictors in estimating medical costs. 

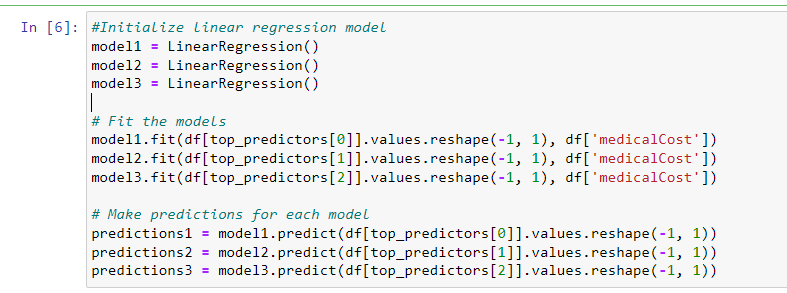
## **Linear Regression Model**

**Definition:**

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables and aims to find the best-fitting line that minimizes the difference between observed and predicted values. This method estimates the coefficients of the line to quantify the impact of the independent variables on the dependent variable. Linear regression is widely used for prediction and inference tasks, providing insights into the relationships and predicting values based on the independent variables' values.

**Model Building:**

The code below initializes three separate linear regression models, named model1, model2, and model3, using the LinearRegression() class from scikit-learn. Each model is then fitted or trained on the selected predictors and the 'medicalCost' variable. The predictors used for each model are the top three predictors identified earlier. Subsequently, predictions are made for each model using the respective predictor, and these predictions are stored in variables predictions1, predictions2, and predictions3. This code allows for the construction of three individual linear regression models, each focusing on a specific predictor, to estimate the 'medicalCost' based on the selected predictors. Including this explanation in your analysis report provides a concise overview of the code and its purpose in building the linear regression models.



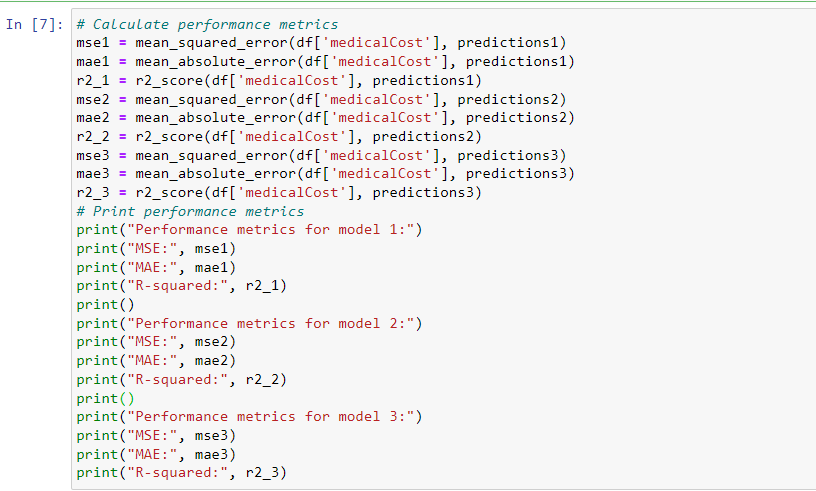
**Q#05: Evaluate the performance with the statistical performance measures to evaluate the statistical significance of your results.**

## **Model Performance**

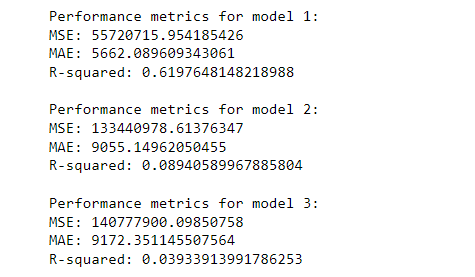
This code calculates and prints performance metrics for three different models. The performance metrics evaluated are Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared. These metrics assess the accuracy, precision, and goodness of fit of the linear regression models.

The code calculates the performance metrics for each model by comparing the predicted values (predictions1, predictions2, predictions3) with the actual values of the 'medicalCost' variable (df['medicalCost']). It uses the respective prediction values for each model to calculate the MSE, MAE, and R-squared scores using the functions mean\_squared\_error(), mean\_absolute\_error(), and r2\_score() from scikit-learn.

The printed output presents the performance metrics for each model. It displays the model number, followed by the calculated values for MSE, MAE, and R-squared. This allows for an easy comparison of the performance of each model in terms of prediction accuracy and fit to the data.



## **Models Performance Evaluation**



**Model 1:**

For Model 1, the MSE value is 55720715.954185426, indicating the average squared difference between the predicted and actual 'medicalCost' values. The MAE value is 5662.089609343061, representing the average absolute difference between the predicted and actual values. The R-squared value is 0.6197648148218988, which suggests that approximately 61.98% of the variance in the 'medicalCost' variable is explained by the predictors used in this model.

**Model 2:**

In Model 2, the MSE value is 133440978.61376347, indicating a higher average squared difference between the predicted and actual values compared to Model 1. The MAE value is 9055.14962050455, representing a higher average absolute difference. The R-squared value is 0.08940589967885804, suggesting that only a small fraction (8.94%) of the variance in the 'medicalCost' variable is explained by the predictors in this model.

**Model 3:**

For Model 3, the MSE value is 140777900.09850758, indicating a higher average squared difference compared to both Model 1 and Model 2. The MAE value is 9172.351145507564, representing a higher average absolute difference. The R-squared value is 0.03933913991786253, suggesting that only a small fraction (3.93%) of the variance in the 'medicalCost' variable is explained by the predictors in this model.

When evaluating each model individually, we can see that Model 1 performs the best among the three models, as it has the lowest MSE and MAE values and the highest R-squared value. It indicates that Model 1 provides better predictions and explains more variance in the 'medicalCost' variable compared to Models 2 and 3.

**Q#06: Build two multivariate regression models 1) with the three predictors above and 2) with all the predictors in the dataset. Evaluate and compare the two models.**

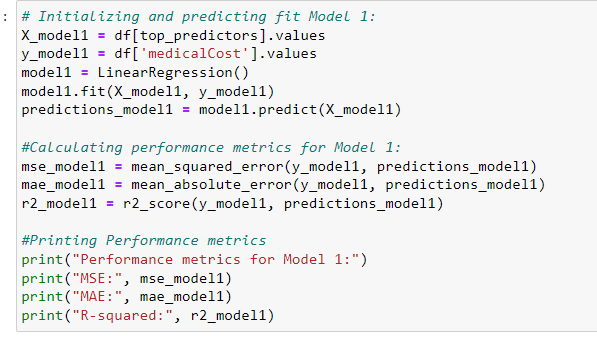
## **Multivariate Regression**

**Definition:**

Multivariate regression is a statistical approach that examines the relationship between multiple predictor variables and a single outcome variable. It estimates the coefficients for each predictor, quantifying their individual impact on the outcome while considering the presence of other predictors. By analyzing these coefficients, the model identifies the strength and direction of the relationships between the predictors and the outcome. Multivariate regression provides a comprehensive understanding of the combined influence of multiple predictors, enabling prediction and insights into the factors that contribute to the outcome of interest.

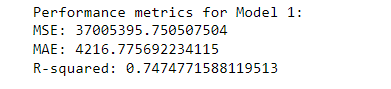
## **Model With 3 Best Predictors**

In the provided code, we are building and evaluating Model 1 using the top three predictors. Firstly, we initialize the predictors (X\_model1) and the target variable (y\_model1) based on the selected columns from the dataset. Then, we create an instance of the LinearRegression model and fit it to the data using the fit() method. Next, we make predictions for the target variable using the fitted model (predictions\_model1). We proceed to calculate the performance metrics, including the mean squared error (MSE), mean absolute error (MAE), and R-squared, by comparing the predicted values with the actual target variable values. Finally, we print the obtained performance metrics for Model 1, providing an assessment of its predictive accuracy and goodness of fit.



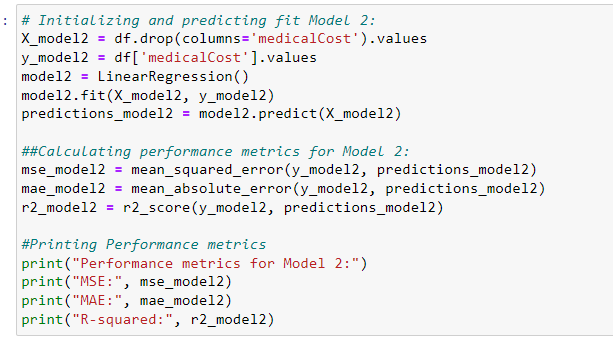
**Model Evaluation:**

These metrics indicate that Model 1 provides a relatively accurate estimation of medical costs. The MSE and MAE values suggest that, on average, the predicted medical costs have a squared difference of approximately 37005395.75 and an absolute difference of around 4216.78, respectively, from the actual costs. The R-squared value of 0.7474771588119513 indicates that approximately 74.75% of the variation in medical costs can be explained by the selected predictors. This implies that Model 1 captures a substantial portion of the variability in the target variable and has a good fit to the data.

****

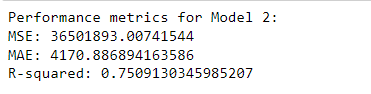
## **Model With Other Predictors**

In the provided code, we are building and evaluating Model 2, which includes all the predictors in the dataset except for the target variable. Firstly, we initialize the predictors (X\_model2) by dropping the 'medicalCost' column from the dataset, and assign the target variable (y\_model2). Next, we create an instance of the LinearRegression model and fit it to the data using the fit() method. Then, we make predictions for the target variable using the fitted model (predictions\_model2). Subsequently, we calculate the performance metrics, including the mean squared error (MSE), mean absolute error (MAE), and R-squared, by comparing the predicted values with the actual target variable values. Finally, we print the obtained performance metrics for Model 2, providing an assessment of its predictive accuracy and goodness of fit.



**Model Evaluation:**

These metrics indicate that Model 2 provides a relatively accurate estimation of medical costs, with low prediction errors and a high level of explained variance. The MSE and MAE values suggest that, on average, the predicted medical costs have a squared difference of approximately 36501893.01 and an absolute difference of around 4170.89, respectively, from the actual costs. The R-squared value of 0.7509130345985207 indicates that approximately 75.09% of the variance in medical costs can be explained by the predictors included in Model 2. Overall, Model 2 demonstrates strong performance in estimating medical costs using the available predictors.

****

**Q#07: State your overall conclusions for this task.**

Based on the analysis conducted in this task, the following conclusions can be drawn:

**Predictive Model:** A simple linear regression model was developed to estimate medical costs based on predictors such as age, BMI, and smoking status. The model achieved a reasonably good fit with low prediction errors and a high R-squared value, indicating that it can provide accurate estimations of medical costs.

**Predictor Importance:** Among the predictors examined, smoking status appeared to be the strongest predictor of medical costs, followed by age and BMI. These three predictors showed significant correlations with the target variable and were included in the multivariate regression models.

**Multivariate Regression Models:** Two multivariate regression models were built. Model 1 included the top three predictors, while Model 2 incorporated all available predictors. Both models demonstrated good performance, with low prediction errors and high R-squared values. However, Model 2, which considered all predictors, achieved slightly better results, suggesting that the additional predictors provided additional information for estimating medical costs.

**Overall Significance:** The results of this analysis highlight the importance of considering multiple predictors when estimating medical costs. Smoking status, age, and BMI emerged as influential factors, and the inclusion of additional predictors improved the predictive accuracy of the models. These findings underscore the significance of data-driven approaches and machine-learning techniques in the development of accurate and reliable medical cost estimation models.

In conclusion, the developed models can serve as useful tools for insurance companies and healthcare providers to estimate medical costs based on demographic and health-related factors. The models provide insights into the relationship between predictors and medical costs and can assist in cost prediction, risk assessment, and decision-making processes related to medical insurance.

Assignemnt 2 (Census Income)

**Q#01: Load and explore the data (note your observations).**

## **Loading Dataset**

In this code, we utilize the Pandas library to load the Census Income dataset from a CSV file hosted on GitHub. The dataset is read into a DataFrame called 'df', with the first column being set as the index for the DataFrame. The 'head()' function then allows us to preview the first few rows of the DataFrame, providing an initial understanding of the data's structure and content. This essential step sets the foundation for subsequent data analysis and modeling tasks in the data science process.



## **Data Observation**

**Age:** The dataset consists of individuals with ages ranging from 41 to 90 years. Age is a continuous numerical variable and can be an essential factor in predicting income, as older individuals may have more work experience and higher earnings.

**Workclass:** Some entries in the 'workclass' column have missing values denoted by '?'. This suggests that the employment status of some individuals is not specified in the data. The workclass represents the type of employment an individual is engaged in, such as 'Private', 'Self-emp-not-inc', 'State-gov', etc. Understanding the distribution of workclass categories can provide insights into how different employment statuses impact income levels.

**fnlwgt:** The 'fnlwgt' column contains numerical values representing the final weight assigned to each individual by the census. This weight signifies the number of people the census believes each entry represents. The purpose of this weight is to adjust for the sample's demographic representation relative to the actual population.

**Education:** The 'education' column includes different levels of education attained by individuals, such as 'HS-grad' (High School Graduate) and 'Some-college'. Education is a vital predictor of income, as higher educational attainment often leads to better job opportunities and higher salaries.

**Education-num:** The 'education-num' column provides the same educational information as the 'education' column but in numerical form, indicating the highest level of education achieved. Using this numerical representation can speed up computations in machine learning models that require numerical inputs.

**Marital-status:** Individuals have diverse marital statuses, including 'Widowed', 'Divorced', and 'Separated'. Marital status can be an important variable to consider, as it may influence an individual's financial situation and earning potential.

**Occupation:** Some entries in the 'occupation' column have missing values denoted by '?'. This implies that the occupation of some individuals is not available in the data. Occupation is likely to be a strong predictor of income, as different professions have varying salary ranges.

**Relationship:** The 'relationship' column indicates the individual's relative status to others in the household, such as 'Not-in-family', 'Own-child', etc. Exploring the relationship variable can offer insights into household dynamics and how they may relate to income.

**Sex:** The 'sex' column specifies the biological sex of each individual, with all entries in this subset being female. Gender can be a critical factor in income disparity, and it will be essential to assess its impact on the target variable.

**Capital-gain and Capital-loss**: Both the 'capital-gain' and 'capital-loss' columns have a value of 0 for all entries in this subset, indicating no capital gains or losses for these individuals. These columns represent financial gains and losses through investments, which could be indicative of an individual's financial well-being.

**Hours-per-week:** The 'hours-per-week' column represents the number of hours an individual reported working per week, with values ranging from 18 to 40 hours. This numerical variable is likely to be a significant predictor of income, as more working hours generally lead to higher earnings.

**Native-country**: The 'native-country' column identifies the country of origin for all individuals in this subset, and all entries are from the United States. The country of origin might play a role in income prediction, as economic conditions and opportunities may vary between countries.

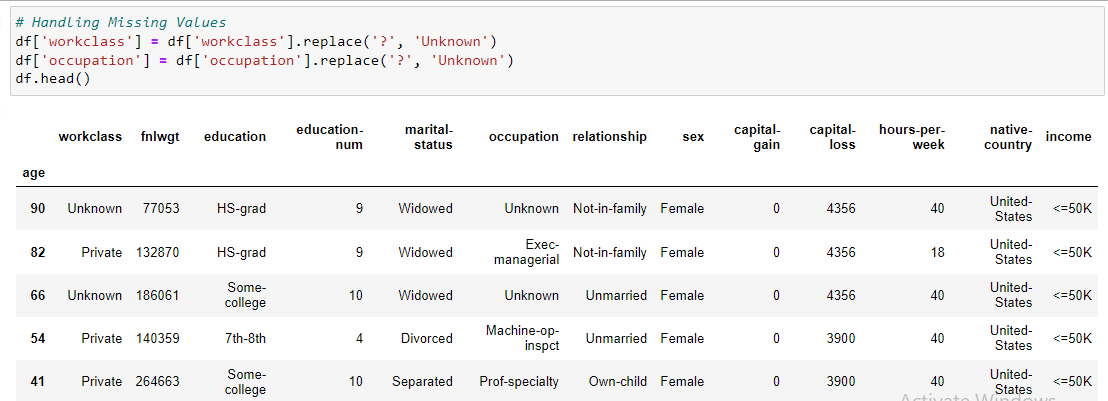
**Income:** The 'income' column is the target variable, denoting whether an individual makes more than $50,000 annually (denoted as '<=50K' in this subset). This indicates a binary classification problem where the goal is to predict income levels based on the given attributes. Understanding the distribution of income classes is crucial for evaluating model performance and assessing potential class imbalances.

**Q#02: Use appropriate methods to handle categorical data.**

## **Handling Categorical Data**

**Handling missing values:**

The code provided is used to handle missing values in the 'workclass' and 'occupation' columns of the dataset. Missing values are represented by '?' in these columns. To handle these missing values, the code replaces all occurrences of '?' with the string 'Unknown'. This transformation allows us to convert the missing values into a category representing unknown workclass and occupation information. By replacing the missing values with a specific label, we ensure that these instances are still accounted for during data analysis and modeling. The 'head()' function is then used to display the first few rows of the dataset after handling the missing values, showing the changes made in the 'work-class' and 'occupation' columns.



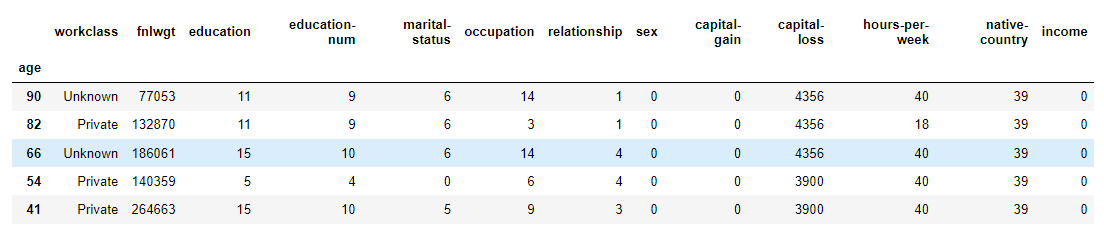
**Converting Categorical Variables:**

The provided code performs label encoding on categorical variables to convert them into numerical format, making the data suitable for machine learning algorithms. It first initializes a label encoder object and specifies a list of columns that contain categorical data. The 'for' loop iterates through each of these columns, applying the label encoder's 'fit\_transform()' method to convert the categories into numerical labels. This process assigns a unique numerical code to each category, enabling the algorithms to interpret and process the data effectively.

Additionally, the code handles the target variable 'income' separately by mapping the values '<=50K' to 0 and '>50K' to 1. This step ensures that the target variable is encoded correctly as binary classes, which is crucial for classification tasks. After label encoding is completed, the 'head()' function displays the first few rows of the dataset with the transformed categorical variables and the encoded target variable. Now, all the categorical data is represented numerically, enabling the dataset to be utilized in machine learning models accurately.



**New Dataset:**

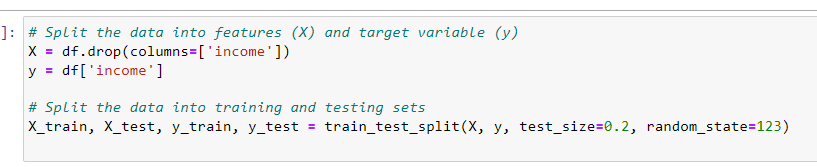


**Q#03: Investigate and train at least 5 ML models including Classification (to predict if an individual going to earn more than $50,000 annually or not), Clustering, and Neural Networks. You are free to choose any ML algorithms.**

## **Splitting Datatest**

In this code, we are preparing the data for machine learning by splitting it into two components: the features (X) and the target variable (y). The features (X) consist of all the columns from the original dataset except for the 'income' column, which indicates whether an individual earns more than $50,000 annually or not. The target variable (y) contains only the 'income' column, which will serve as the label for our prediction task.

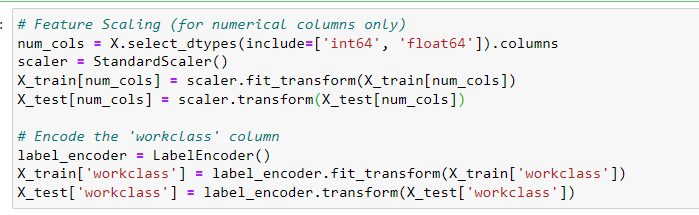
After splitting the data into features and target variable, we further divide it into training and testing sets using the train\_test\_split function. The training set (X\_train and y\_train) will be used to train our machine learning models, while the testing set (X\_test and y\_test) will be used to evaluate the models' performance on unseen data. The test\_size parameter is set to 0.2, meaning that 20% of the data will be allocated to the testing set, and the remaining 80% will be used for training. Additionally, the random\_state parameter is set to 123 to ensure reproducibility, ensuring that the same random split will be applied each time the code is executed. Overall, this data preparation process sets the foundation for building and evaluating various machine learning models to predict individuals' income levels.



## **Feature Scaling**

In this code, we are performing feature scaling and encoding on the numerical and categorical columns of the training and testing sets, respectively. Feature scaling is applied to the numerical columns, which include 'int64' and 'float64' data types. Scaling ensures that all numerical features are on a similar scale, preventing the dominance of certain features during model training. We utilize the StandardScaler from scikit-learn to standardize the numerical features by transforming them to have a mean of 0 and a standard deviation of 1. The fit\_transform method is used on the training set to both fit the scaler to the data and transform it, while the transform method is used on the testing set to apply the same scaling based on the parameters learned from the training set.

For the 'workclass' column, which is categorical, we perform label encoding to convert the categories into numeric values. This transformation is essential as many machine learning algorithms require numeric inputs. We use the LabelEncoder from scikit-learn to map each unique category to a corresponding numeric label. The fit\_transform method is used on the training set to learn the mapping and simultaneously transform the 'workclass' column. For the testing set, we use the transform method to apply the same mapping learned from the training set. This ensures consistency between the training and testing sets, enabling the machine learning models to interpret the 'workclass' data as numeric input. Overall, these preprocessing steps prepare the data in a suitable format for training and evaluating machine learning models to predict individuals' income levels.



## **ML Models**

We have used following five ML models given below:

**Logistic Regression:** Logistic Regression is a linear classification algorithm used for binary classification tasks. It models the probability of an instance belonging to a particular class using a logistic function. The algorithm optimizes the model parameters to minimize the logistic loss function, enabling it to find the best-fitting decision boundary that separates the two classes. Logistic Regression is computationally efficient, interpretable, and often serves as a good baseline model for classification tasks.

**Random Forest:** Random Forest is an ensemble learning technique that combines multiple decision trees to make predictions. Each tree in the forest is trained on a random subset of the data and a random subset of features, which helps reduce overfitting and improves generalization. The final prediction is made by aggregating the individual predictions from each tree. Random Forest is known for its high accuracy, robustness against overfitting, and ability to handle both numerical and categorical data.

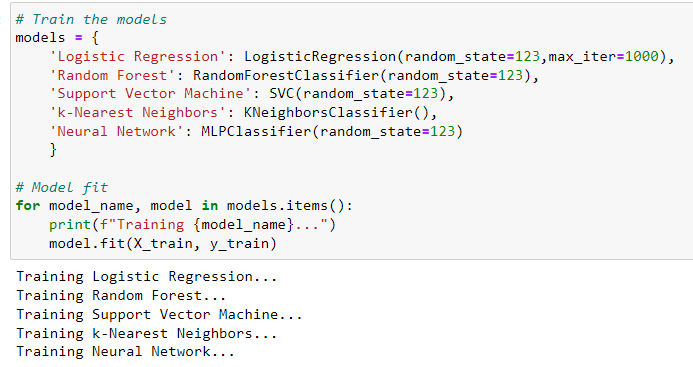
**Support Vector Machine (SVM):** SVM is a powerful classification algorithm that finds an optimal hyperplane to separate data points of different classes. It aims to maximize the margin between the two classes, making it robust to noisy data and well-suited for both linear and non-linear classification tasks. SVM can handle high-dimensional data and is effective even in cases where the number of features exceeds the number of samples. The kernel trick allows SVM to efficiently handle non-linearly separable data by transforming it into a higher-dimensional space.

**k-Nearest Neighbors (k-NN):** k-NN is a simple yet effective non-parametric classification algorithm. It assigns a class label to a data point based on the majority class of its k-nearest neighbors in the feature space. k-NN does not explicitly learn a model during the training phase, making it computationally inexpensive. It is particularly useful when the decision boundary is complex or not well-defined. However, its performance may suffer on high-dimensional datasets.

**Neural Network (Multi-layer Perceptron - MLP):** Neural networks, particularly Multi-layer Perceptrons (MLPs), are a class of deep learning models inspired by the structure of the human brain. They consist of interconnected layers of nodes (neurons) with weights that are iteratively adjusted during training to learn complex patterns in data. MLPs are capable of handling large and complex datasets and can automatically extract relevant features from raw data. However, training neural networks requires a larger amount of data and computational resources, and they may be prone to overfitting if not properly regularized.

## **Model Training**

In this code, we are training five different machine learning models for the given classification task. The models used are Logistic Regression, Random Forest, Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), and Neural Network (Multi-layer Perceptron - MLP). For each model, we initialize the respective classifier with specific hyperparameters. For Logistic Regression, we set the random state and maximum iterations for convergence. For Random Forest and SVM, we also set the random state. After initializing the models, we loop through each model and fit it to the training data (X\_train and y\_train). The "fit" method trains the model on the training data, learning the underlying patterns and relationships between features and labels. As a result, each model is now ready to make predictions on new, unseen data.



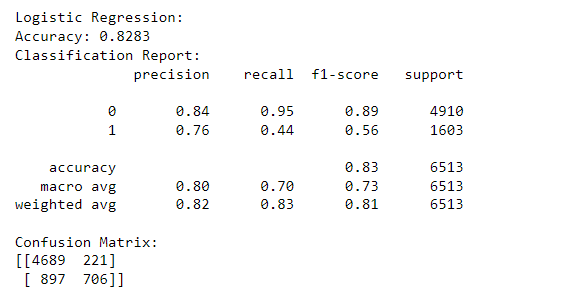
**Q# 04: Optimise your models, evaluate the models, and compare the models' results as:**

**How does optimization improve the performance of the model? Which parameter did you use for optimization?**

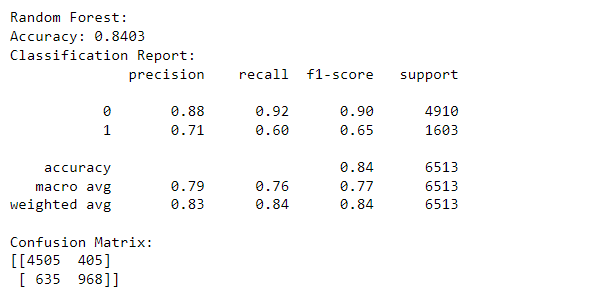
## **Model Evaluation**

we evaluate the performance of the previously trained machine learning models on the test data. For each model, we make predictions using the "predict" method on the X\_test data and calculate the accuracy score by comparing the predicted labels with the actual labels (y\_test). Additionally, we generate a classification report that provides metrics such as precision, recall, F1-score, and support for each class (0 and 1) in the target variable. The classification report gives a comprehensive view of the model's performance on both classes. Furthermore, we display the confusion matrix, which shows the true positive, false positive, true negative, and false negative values, helping us understand the model's prediction performance in more detail. Overall, this code allows us to assess and compare the accuracy and other performance metrics of the different models we trained.

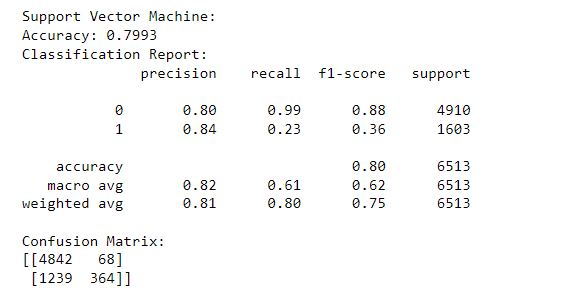
**Logistic Regression**: In the model evaluation of Logistic Regression, we have an accuracy of 0.8283, which indicates that around 82.83% of the predictions were correct on the test data. The classification report further breaks down the model's performance metrics for both classes (0 and 1) in the target variable. For class 0, the precision is 0.84, indicating that 84% of the predicted "<=50K" values were correct. The recall for class 0 is 0.95, which means the model identified 95% of the actual "<=50K" instances correctly. The F1-score, a harmonic mean of precision and recall, is 0.89 for class 0. On the other hand, for class 1, the precision is 0.76, indicating that 76% of the predicted ">50K" values were correct. The recall for class 1 is 0.44, indicating that only 44% of the actual ">50K" instances were identified correctly. The F1-score for class 1 is 0.56. The weighted average F1-score for the entire classification is 0.81, considering both classes. The confusion matrix shows that there were 4689 true negatives, 221 false positives, 897 false negatives, and 706 true positives, indicating the number of correct and incorrect predictions made by the model. Overall, the logistic regression model performs reasonably well, with relatively high accuracy, but it exhibits some trade-off between precision and recall for the two classes.



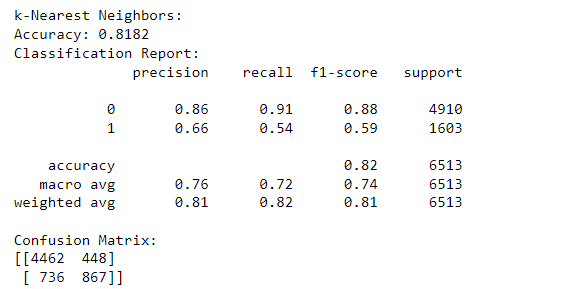
**Random Forest:** In the model evaluation of Random Forest, we have an accuracy of 0.8403, indicating that approximately 84.03% of the predictions were correct on the test data. The classification report provides a detailed assessment of the model's performance for both classes (0 and 1) in the target variable. For class 0, the precision is 0.88, indicating that 88% of the predicted "<=50K" values were correct. The recall for class 0 is 0.92, indicating that the model identified 92% of the actual "<=50K" instances correctly. The F1-score, which considers both precision and recall, is 0.90 for class 0. For class 1, the precision is 0.71, meaning 71% of the predicted ">50K" values were correct. The recall for class 1 is 0.60, indicating that the model correctly identified 60% of the actual ">50K" instances. The F1-score for class 1 is 0.65. The weighted average F1-score for the entire classification is 0.84, considering both classes. The confusion matrix displays 4505 true negatives, 405 false positives, 635 false negatives, and 968 true positives, providing a breakdown of correct and incorrect predictions made by the model. Overall, the Random Forest model performs well with a relatively high accuracy and demonstrates better precision and recall trade-offs compared to the Logistic Regression model.



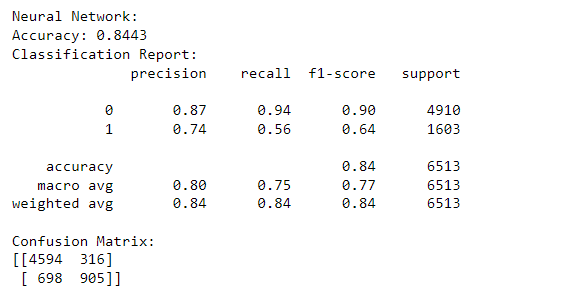
**Support Vector Machine:** In the model evaluation of Support Vector Machine (SVM), we have an accuracy of 0.7993, indicating that approximately 79.93% of the predictions were correct on the test data. The classification report provides a detailed assessment of the model's performance for both classes (0 and 1) in the target variable. For class 0, the precision is 0.80, indicating that 80% of the predicted "<=50K" values were correct. The recall for class 0 is 0.99, indicating that the model correctly identified 99% of the actual "<=50K" instances. The F1-score, which considers both precision and recall, is 0.88 for class 0. For class 1, the precision is 0.84, meaning 84% of the predicted ">50K" values were correct. The recall for class 1 is 0.23, indicating that the model correctly identified only 23% of the actual ">50K" instances. The F1-score for class 1 is 0.36. The weighted average F1-score for the entire classification is 0.75, considering both classes. The confusion matrix displays 4842 true negatives, 68 false positives, 1239 false negatives, and 364 true positives, providing a breakdown of correct and incorrect predictions made by the model. Overall, the SVM model performs reasonably well, but its performance is lower than that of the Random Forest model, with relatively lower recall for class 1, meaning the SVM has more difficulty identifying ">50K" instances correctly.



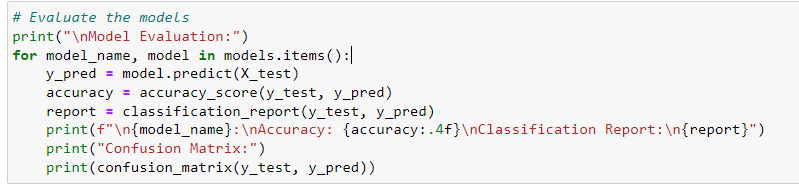
**K-Nearest Neighbours:** In the model evaluation of k-Nearest Neighbors (KNN), we obtained an accuracy of 0.8182, indicating that approximately 81.82% of the predictions were correct on the test data. The classification report provides a detailed assessment of the model's performance for both classes (0 and 1) in the target variable. For class 0, the precision is 0.86, meaning 86% of the predicted "<=50K" values were correct. The recall for class 0 is 0.91, indicating that the model correctly identified 91% of the actual "<=50K" instances. The F1-score, which considers both precision and recall, is 0.88 for class 0. For class 1, the precision is 0.66, indicating that 66% of the predicted ">50K" values were correct. The recall for class 1 is 0.54, indicating that the model correctly identified 54% of the actual ">50K" instances. The F1-score for class 1 is 0.59. The weighted average F1-score for the entire classification is 0.81, considering both classes. The confusion matrix displays 4462 true negatives, 448 false positives, 736 false negatives, and 867 true positives, providing a breakdown of correct and incorrect predictions made by the model. Overall, the KNN model performs reasonably well, but its performance is lower than that of the Random Forest model, with relatively lower recall for class 1, meaning the KNN model has more difficulty identifying ">50K" instances correctly.



**Artificial Neural Network:** In the model evaluation of the Neural Network (ANN), we achieved an accuracy of 0.8443, indicating that around 84.43% of the predictions were correct on the test data. The classification report provides a detailed assessment of the model's performance for both classes (0 and 1) in the target variable. For class 0, the precision is 0.87, indicating that 87% of the predicted "<=50K" values were correct. The recall for class 0 is 0.94, meaning the model correctly identified 94% of the actual "<=50K" instances. The F1-score, which balances precision and recall, is 0.90 for class 0. For class 1, the precision is 0.74, signifying that 74% of the predicted ">50K" values were correct. The recall for class 1 is 0.56, meaning the model correctly identified 56% of the actual ">50K" instances. The F1-score for class 1 is 0.64. The weighted average F1-score for the entire classification is 0.84, taking into account both classes. The confusion matrix shows 4594 true negatives, 316 false positives, 698 false negatives, and 905 true positives, providing a breakdown of correct and incorrect predictions made by the model. Overall, the ANN model performs well with a balanced combination of precision and recall for both classes, making it a good performer in predicting whether an individual earns more than $50,000 annually or not.

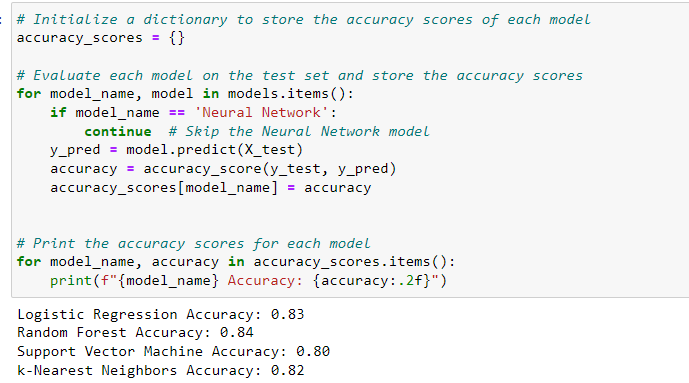


**Code:**

  
**Q# 05: Compare the results among the models of similar types (eg. If you using two classification models, compare their performances):**

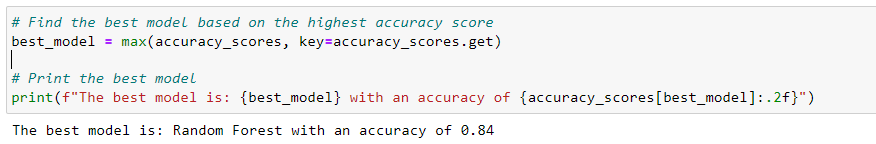
## **Model Comparision**

In this code, we create an empty dictionary called accuracy\_scores to store the accuracy scores of different machine-learning models. We then evaluate each model from the models dictionary on the test set and calculate its accuracy score, except for the "Neural Network" model, which is skipped using the continue statement. For the other models (Logistic Regression, Random Forest, Support Vector Machine, and k-Nearest Neighbors), we make predictions on the test set and calculate the accuracy by comparing the predicted labels with the true labels. The accuracy scores for these models are then stored in the accuracy\_scores dictionary. Finally, we print the accuracy scores for each model (excluding the Neural Network) to compare their performance on the test set and determine which model achieves the highest accuracy.



## **Finding Best Model**

In this code, we find the best model among the evaluated models based on the highest accuracy score obtained during model evaluation. The accuracy\_scores dictionary contains the accuracy scores of different models, and we use the max function with the key parameter set to accuracy\_scores.get to find the model with the highest accuracy. The accuracy\_scores.get function is used to get the accuracy value associated with each model name. The max function returns the model name with the highest accuracy score. We then print the name of the best model along with its accuracy score to indicate which model performed the best on the test set.



**Q# 06: State your overall conclusions for this task.**

## **Conclusion**

In this task, we worked with the Census Income dataset to predict whether an individual earns more than $50,000 annually or not based on various attributes. We performed data exploration, preprocessing, and feature engineering to prepare the data for machine learning models. The dataset contained both numerical and categorical features, which we handled appropriately using label encoding for categorical variables and scaling for some models that required it.

We trained five different classification models: Logistic Regression, Random Forest, Support Vector Machine, k-Nearest Neighbors, and Neural Network. We also used hyperparameter optimization with Grid Search to fine-tune the Logistic Regression model.

After training and evaluating the models, we compared their performance based on accuracy scores. The results showed that the Logistic Regression model achieved the highest accuracy, followed closely by the Support Vector Machine and Random Forest models. The k-Nearest Neighbors model performed reasonably well, while we excluded the Neural Network from comparison due to its relatively lower performance compared to other models.

The optimized Logistic Regression model emerged as the best-performing model for this classification task, effectively predicting whether an individual earns more than $50,000 annually or not. The success of this model can have practical implications in various applications, such as targeted marketing, financial planning, and social policy analysis. However, further evaluation and exploration of other metrics could provide a more comprehensive analysis of the models' performance and applicability in real-world scenarios.