Income Prediction

● Introduction:

The project aims to predict income levels based on various demographic and employment-related features using machine learning algorithms. We explore different models, evaluate their performance, and provide insights into the factors that contribute to higher income levels. The importance of this project lies in, for example, helping non-profit organizations evaluate their much-needed donation requests from different individuals.

● The Dataset:

The dataset provided to us contains 32560 rows, and 14 different independent features. We aim to predict if a person earns more than 50k$ per year or not. Since the data predicts 2 values (>50K or <=50K), this clearly is a classification problem, and we will train the classification models to predict the desired outputs. Mentioned below are the details of the features provided to us, which we will be feeding to our classification model to train it.

1. Age: The age of an individual, this ranges from 17 to 90.
2. Workclass: The class of work to which an individual belongs.
3. Fnlwgt: The weight assigned to the combination of features (an estimate of how many people belong to this set of combination)
4. Education: Highest level of education
5. Education_num: Number of years for which education was taken.
6. Marital_Status: Represents the category assigned based on marriage status of a person.
7. Occupation: Profession of a person
8. Relationship: Relation of the person in his family
9. Race: Origin background of a person
10. Sex: Gender of a person
11. Capital_gain: Capital gained by a person.
12. Capital_loss: Loss of capital for a person.
13. Hours_per_week: Number of hours for which an individual works per week
14. Native_Country: Country to which a person belongs.

Output:

Income — The target variable, which predicts if the income is higher or lower than 50K$.

● EDA:

First: We saw an overview to our Train data, how many rows and columns we have, and what are the data types of our columns.
Second: We make **Data Cleaning;** we didn't find nulls and we found outliers, but we tried to handle it or keep it as it is, we found that the accuracy was better when we keep it. We dropped "Education" column because it gives the same information as "Education_num". And we dropped 'fnlwgt' as it is going to confuse our model. We Checked for duplicates and removed it. And There was a wrong data '?', we looked at their percentage we found it very small, so we replaced it by mode.
Third: We make **Data Visualization;** we make a pair plot to look on all the columns, then we make a graph for each column to see the count of each value and how it is affecting our target column.
Fourth: We make **Data Preprocessing;** we make encoding for the categorical columns using label encoding and we make standardization, but it decreased the accuracy, so we removed it.
Fifth: We make **Feature selection;** we reduced the values in marital-status to be easy for the model. We used chi-square and ANOVA tests to see the correlation between columns and what are the columns affecting our target column.
Sixth: We clean the **Test Data** and prepare it for prediction.

● Modeling:

- **Logistic Regression**: is a statistical model used for binary classification problems, where the goal is to predict the probability of an event or outcome belonging to one of two classes. It is a popular and widely used algorithm in machine learning and statistics. The logistic regression model uses the logistic function (also known as the sigmoid function) to model the relationship between the features (input variables) and the probability of the outcome. The logistic function maps any real-valued number to a value between 0 and 1, representing the probability.
- **SVM:** The SVC (Support Vector Classifier) is a popular algorithm used for classification tasks in machine learning. It belongs to the family of supervised learning algorithms and is based on the concept of support vector machines (SVMs). The SVC algorithm works by finding an optimal hyperplane in a high-dimensional feature space that separates the data points of different classes. The goal is to maximize the margin between the classes, which is the distance between the hyperplane and the closest data points of each class.
- **Decision Tree**: It is a non-parametric supervised learning method that builds a flowchart-like structure, where each internal node represents a feature or attribute, each branch represents a decision rule, and each leaf node represents the outcome or prediction. The decision tree algorithm recursively partitions the training data based on the values of different features, aiming to create homogeneous subsets at each level that are as pure as possible in terms of the target variable (for classification tasks). This partitioning process continues until a stopping criterion is met, such as reaching a maximum depth or a minimum number of samples in a leaf node.

- **Random Forest**: Random Forest is a popular ensemble learning algorithm that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It is widely used for both classification and regression tasks in machine learning. The Random Forest algorithm works by creating a collection of decision trees, where each tree is trained on a different subset of the training data, randomly selected with replacement. Additionally, at each split in the tree, only a random subset of features is considered for determining the best split.
- **XGboost**: is designed to handle both classification and regression tasks and can work with a variety of data types, including numerical and categorical features. It builds an ensemble of weak prediction models, typically decision trees, in a sequential manner, where each new model corrects the mistakes made by the previous models.

These was the results we get after using the models:

	Accuracy	**F1**
Logistic Regression	81.89	0.50
SVM	80.10	0.39
Decision Tree	85.56	0.65
Random Forest	85.57	0.66
XGboost	86.56	0.69

As we see XGboost give us the best accuracy and best f1. So, we decided to tune some hyperparameters to get a greater accuracy.

1. **n_estimators:** It refers to the number of boosting rounds or decision trees to be built. Increasing the number of estimators may improve the model's performance, but it can also lead to overfitting if set too high. It's a crucial parameter to tune as it balances model complexity and computation time.
2. **learning_rate:** It determines the step size at each boosting iteration. A smaller learning rate makes the model converge slowly but can potentially yield better results by allowing more fine-grained adjustments. On the other hand, a larger learning rate makes the model converge faster but may result in overshooting the optimal solution.
3. **max_depth:** It sets the maximum depth of each decision tree. A higher value allows the tree to capture more complex interactions in the data, but it can also lead to overfitting. It's important to find an optimal value that balances the model's capacity to learn complex patterns without compromising generalization.
4. **gamma:** It specifies the minimum loss reduction required to split a node further. A higher gamma value makes the algorithm more conservative, as it requires a higher loss reduction to make additional splits. This parameter can help control the complexity of the tree and prevent overfitting.

● Conclusion: By tuning these ones and see the effect on our accuracy we get that when n_estimators=1000, learning_rate=0.04, max_depth=5, gamma=0.11

It gives us the highest accuracy: 86.778 %.